Heuristic solution approaches to the fixed charge transportation problem

RP Dikgale

Thesis presented in partial fulfilment of the requirements for the degree of

Master of Commerce (Operations Research)

in the Faculty of Ecomnomic and Management Sciences at Stellenbosch University

Supervisor: Prof SE Visagie

April 2019

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2019

Abstract

The classical transportation problem is concerned with the distribution of a single commodity from a group of supply centres or sources, to a group of demand centres or destinations. The amount of commodity available at any source is limited, and the demand for the commodity at each destination is finite. Transportation cost functions may be non-linear because of quantity discounts, or price breaks, etc. Also, a fixed charge may be incurred every time units of commodity are sent from a given source to a given destination. The fixed charge transportation problems (FCTP) differs from the standard linear transportation problem (TP) only in the nonlinearity (caused by die fixed charge) in the objective function.

Different heuristic methods were developed to generate initial solutions. The stepping stone method and tabu search algorithm are used to attempt to solve this problem. The algorithms are evaluated according to their efficiency (computational runtime and solution quality) for solving FCTP problems. Comparisons are made using randomly generated benchmark instances from the literature. The instances contain different sizes and different ranges of magnitude of fixed costs relative to variable costs. The primal-dual algorithm was also considered in finding good solutions to be FCTP.

The results (for small instances) obtained for the proposed algorithm have been compared with that for an exact algorithm based on an integer programming formulation available in the literature. The results from computational experiments show that the proposed algorithms yield near optimal solution to most instances. The primal-dual algorithm demonstrate significant improvement over the proposed heuristic methods for small FCTPs, although it could not find feasible solutions to some instances.

Opsomming

Die klassieke vervoerprobleem ondersoek die verspreiding van een soort gebruiksartikel vanaf 'n groep verskafferpunte of bronne na 'n groep aanvraagpunte of bestemmings. Die hoeveelheid van die gebruiksartikels beskikbaar by elke bron is beperk en die aanvraag by elke bestemming is eindig. Die vervoerkostefunksie mag nielineêr wees as gevolg van grootmaatafslag, pryspunte ens. 'n Vaste koste kan ook gehef word elke keer wanneer gebruiksartikels van 'n gegewe bron na 'n gegewe bestemming vervoer word. Hierdie vaste koste vervoerprobleem (FCTP) verskil van die standaard lineêre vervoerprobleem (TP) slegs in die nie-lineariteit (as gevolg van die vaste koste) in die doelfunksie.

Verskillende heuristieke word voorgestel om beginoplossings te genereer. Die kringloopmetode saam met 'n tabusoektog word gebruik in 'n poging om hierdie probleem op die los. Die algoritmes word geëvalueer in terme van hul effektiewiteit (berekeningstyd en oplossingskwaliteit). Die vergelykings word gemaak met lukraak gegenereerde probleme uit die literatuur. Hierdie gegenereerde probleme bevat verskillende groottes en verskillende verhoudings van vaste koste tot veranderlike koste. 'n Primaal-duaalalgoritme word ook aangebied om goeie oplossing vir die FCTP te vind.

Die resultate (vir klein voorbeelde) wat vir die voorgestelede algoritmes verkry is, word vergelyk met dié van die eksakte oplossing wat met 'n heeltallige programmeringsformulering beskikbaar in die literatuur verkry is. Die resultate wys dat die algoritmes in die meeste gevalle oplossings na-aan optimaal kry. Die primaal-duaalalgoritme verkry goeie verbeterings op die voorgestelede heuristieke vir FCTP's, maar kon nie in al die gevalle toelaatbare oplossings opspoor nie.

Acknowledgements

The author wishes to acknowledge the following people for their various contributions towards the completion of this work:

- Prof. Stephan Visagie
- Mr Tedros Weldemicael
- Mr Pierre Baard
- Mr Pieter DeWet

Contents

1 Introduction **1**

1.1 Modelling the transportation problem 2

1.2 Purpose of transportation modelling 4

1.3 The nature of transportation costs 4

1.4 Background on transportation and fixed charge problem 4

1.5 Problem description 5

1.6 Aim and objectives 5

1.6.1 Aim of the study 6

1.6.2 Objectives of the study 6

1.7 Significance of the study 6

1.8 Thesis outline 6

2 Literature survey **7**

2.1 Exact solution methods 7

2.2 Heuristic solution methods 10

2.3 Chapter summary 13

3 Data **15**

3.1 Description of the benchmark instances 15

3.2 Linear formulation 17

3.3 Implementation 18

3.4 Chapter summary 18

4 Model **19**

4.1 Mathematical model and descriptions 20

4.1.1 Existence of feasible solution 23

4.1.2 Basic feasible solution 23

4.2 Tabu search algorithm 24

4.3 The initial solution . . . 26
4.3.1 Heuristics methods of finding initial solutions . . . 26
4.3.2 Computational results of initial solution methods . . . 29
4.4 Solution improvement . . . 31
4.4.1 Stepping stone method . . . 31
4.4.2 Selection criteria . . . 33
4.5 Tabu list . . . 35
4.6 Aspiration criteria . . . 35
4.7 A primal-dual method for solving transportation problems . . . 36
4.8 Chapter summary . . . 39

5 Computational experiments **41**
5.1 Stepping stone method with a greedy local search . . . 42
5.2 Computational results on different selection criteria . . . 43
5.3 Stepping stone method with tabu search algorithm . . . 45
5.4 Lingo performance analysis . . . 47
5.5 Experimental analysis on Dataset 2 . . . 48
5.6 The primal-dual algorithm solution experiments . . . 50
5.6.1 Initial feasible solution . . . 50
5.6.2 Solution improvement . . . 50
5.7 Chapter summary . . . 52

6 Conclusion and recommendations **53**
6.1 Thesis summary . . . 53
6.2 Recommendation . . . 54
6.3 Achievement of objectives . . . 54
6.4 Future work . . . 54

A Computational results **55**

List of Figures

1.1 Transportation model with m sources and n destinations. 3

3.1 Shipping cost as a function of quantity shipped along route (i, j) for FCTP. . . . 18

4.1 Flow chart of the fixed charge transportation problem approach 20
4.2 Initial solutions of different heuristic criteria on Dataset 1. 30
4.3 The flow chart showing the improvement process using the stepping stone method 31
4.4 Flow chart of the row-column random selection criterion 33
4.5 Flow chart of the row-column selection criterion 34
4.6 Flow chart of the sequential selection criterion 34
4.7 Flow chart of the random selection criterion 34

5.1 Graphical presentation of solution improvement using the stepping stone method when the greedy local search is considered 42
5.2 Graphical presentation of solution improvement using the stepping stone method when tabu search algorithm is considered 46
5.3 Solution movement to the fixed charge transportation problem when tabu search algorithm is considered 47

List of Tables

3.1 Summary of Dataset 1 test problems 16
3.2 Summary of Dataset 2 test problems 16

4.1 Degenerate transportation tableau 22
4.2 Average computational time (in seconds) to the initial solutions 30
4.3 Stepping stone loops 32

5.1 Summary of best solution obtained by different selection criteria on Set A1 . . . 43
5.2 Summary of best solution obtained by different selection criteria on Set A2 . . . 44
5.3 Summary of best solution obtained by different selection criteria on Set A3 . . . 44
5.4 Average solution gap to the optimal solution on Dataset 1 problem instances . . 45
5.5 Average computational time to the best obtained solution for Dataset 1 46
5.6 Summary of optimal solutions obtained by Lingo solver on Dataset 1 48
5.7 Average solution gap and computational runtime on Set R1 to R9 of Dataset 2 . 49
5.8 Average solution gap and computational runtime on Set R10 to Set R18 of Dataset 2 49
5.9 Initial solutions of Dataset 1 using the primal-dual method 51
5.10 Best solution using the stepping stone method and tabu search algorithm on primal-dual algorithm solutions 51
5.11 Best solution using the stepping stone method and tabu search algorithm on primal-dual algorithm solutions 52

A.1 Heuristic initial solutions for FCTP on Set A1 of Dataset 1 56
A.2 Heuristic initial solutions for FCTP on Set A2 of Dataset 1 56
A.3 Heuristic initial solutions for FCTP on Set A3 of Dataset 1 57
A.4 Best solutions to the greedy search algorithm on Dataset 1 57
A.5 Best solutions to the tabu search algorithm on Dataset 1 58
A.6 Computational results on Set R1 to Set R3 of Dataset 2. 59
A.7 Computational results on Set R4 to Set R6 of Dataset 2. 60
A.8 Computational results on Set R7 to Set R9 of Dataset 2. 61

A.9 Computational results on Set R10 to Set R12 of Dataset 2. 62
A.10 Computational results on Set R13 to Set R15 of Dataset 2. 63
A.11 Computational results on Set R16 to Set R18 of Dataset 2. 64

List of Algorithms

1 Tabu search algorithm 25
2 Random selection criterion 27
3 Relaxed minimum cost selection criterion 28
4 Random minimum cost selection criterion 28
5 Maximum flow minimum cost selection criterion 29
6 Stepping stone algorithm 32

CHAPTER 1

Introduction

Contents

1.1 Modelling the transportation problem 2
1.2 Purpose of transportation modelling 4
1.3 The nature of transportation costs 4
1.4 Background on transportation and fixed charge problem 4
1.5 Problem description 5
1.6 Aim and objectives 5
1.6.1 Aim of the study *6*
1.6.2 Objectives of the study *6*
1.7 Significance of the study 6
1.8 Thesis outline 6

Business and industry are both interested in becoming more competitive and thus invest in things such as cost minimisation. This is essential to the existence of firms. One of these costs is the minimisation of transportation costs. Transportation problems are primarily concerned with the optimal (best possible) way in which a product produced at different factories or plants (called supplies or origins) can be transported to a number of warehouses or customers (called demands or destinations). Whenever there is a physical movement of goods from the point of manufacturer to the final consumers through a variety of channels of distribution (wholesalers, retailers, distributors, etc.), there is a need to minimise the cost of transportation so as to increase profit on sales.

Transportation models deal with the determination of a minimum-cost plan for transporting a single commodity from a number of sources to a number of destinations. The amount of commodity available at any source is limited, and the demand for the commodity at each destination is finite. The commodity being transported need not necessarily be a physical commodity. Furthermore, the transportation itself does not have to involve physical movement. Thus, for example, it is possible to talk of transporting information in the form of data from one computer to another. In production systems, it is possible to model the manufacturing of a product on a set of machines over different time periods as a transportation problem. The terms transportation and commodity are used therefore, in a general sense. It is easiest, however, to conceptualize the problem in the context of physical transportation systems.

Many practical transportation and distribution problems with fixed charges in logistics can be formulated as a fixed charge transportation problem (FCTP). The problem becomes transporting the commodity from the sources to the destinations, so that demand at each destination is met,

without exceeding the available supply at any of the sources. Since the fixed-charge problems was initialized by Hirsch and Dantzig [19], it has been widely applied in many decision making and optimisation problems.

Transportation systems serve people, and are created by people, both the system owners and operators, who run, manage, and maintain the system and travellers who use it. Travellers' time depends both on free flow time, which is a product of the infrastructure design and on delay due to congestion, which is an interaction of system capacity and its use. There also exist the adverse outcomes of transportation. This includes:

- by polluting, systems consume health and increase morbidity and mortality,
- by being dangerous, they consume safety and produce injuries and fatalities,
- by being loud they consume quiet and produce noise (decreasing quality of life and property values), and
- by emitting carbon and other pollutants, they harm the environment.

All of these factors are increasingly being recognized as costs of transportation, but the most notable are the environmental effects, particularly with concerns about global climate change. Transportation is central to economic activity and to people's lives, it enables them to engage in work, attend school, shop for food and other goods, and participate in all of the activities that comprise human existence. More transportation, by increasing accessibility to more destinations, enables people to better meet their personal objectives, but entails higher costs both individually and socially. While the transportation problem is often posed in terms of congestion, that delay is but one cost of a system that has many costs and even more benefits. Further, by changing accessibility, transportation gives shape to the development of land.

1.1 Modelling the transportation problem (TP)

All types of transportation problems can be solved by a general network method, but a specific transportation algorithm is introduced here. The data of the model includes:

- The amount of supply at each source and the amount of demand at each destination, and
- the transportation cost per unit of the commodity from each source to each destination.

Since there is only one commodity, a destination can receive its demand from more than one source. The objective is to determine how much should be shipped from each source to each destination so as to minimise the total transportation cost.

Figure 1.1 graphically demonstrates a transportation model with m sources and n destinations. Each source or destination is represented by a node. The route between a source and destination is represented by an arc joining the two nodes. The amount of supply available at source i is s_i, and the demand required at destination j is d_j. The cost of transporting one unit between source i and destination j is c_{ij}. Let x_{ij} denote the quantity transported from source i to destination j. The cost associated with this movement is cost $\times$ quantity $= c_{ij}x_{ij}$.

The cost of transporting the commodity from source i to all destinations is thus given by

$$\sum_{j=1}^{n} c_{ij}x_{ij} = c_{i1}x_{i1} + c_{i2}x_{i2} + \cdots + c_{in}x_{in}.$$

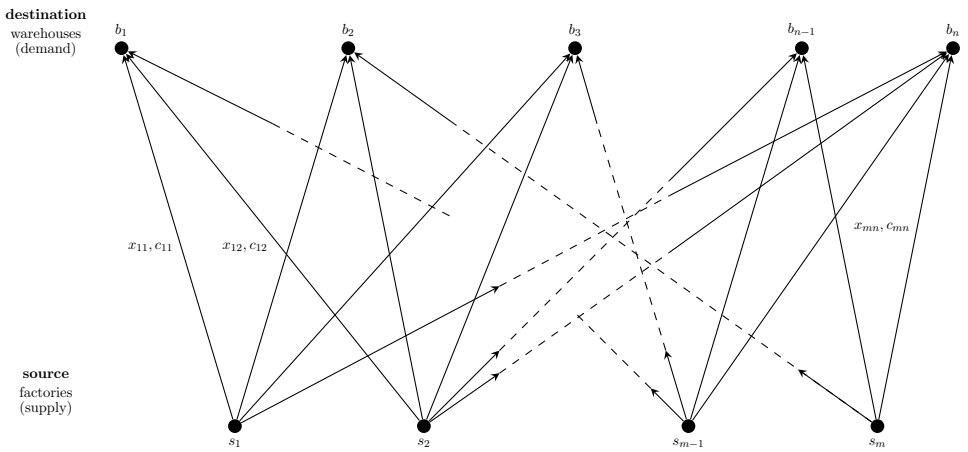


FIGURE 1.1: *Transportation model with m sources and n destinations.*

Thus, the total cost of transporting the commodity from all the sources to all the destinations is

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}.$$

In order to minimise the transportation costs, the following problem must be solved. The objective is to

$$\begin{aligned} \text{minimise} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{n} x_{ij} = s_i \qquad i = 1,\ldots,m, \\ & \sum_{i=1}^{m} x_{ij} = d_j \qquad j = 1,\ldots,n, \\ & x_{ij} \geq 0 \qquad i = 1,\ldots,m; \quad j = 1,\ldots,n. \end{aligned} \tag{1.1}$$

Formulation (1.1) assumes that the total supply ($\sum_{i=1}^{n} s_i$) is equal to the total demand ($\sum_{j=1}^{m} d_j$). When the total supply is equal to the total demand (i.e. $\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j$) then the transportation model is said to be balanced.

Similarly a transportation model in which the total supply and total demand are unequal is called an unbalanced transportation model. It is always possible to balance an unbalanced transportation problem. If total supply exceeds total demand (i.e. $\sum_{i=1}^{n} s_i > \sum_{j=1}^{m} d_j$), the transportation problem can be balanced by creating a dummy demand point that has a demand equal to the amount of excess supply. The shipments to the dummy demand point are not real shipments, thus they are assigned a cost of zero. Shipments to the dummy demand point indicate unused supply capacity. When total supply is less than total demand, it is sometimes desirable to allow the possibility of leaving some demand unmet. In such a situation, a penalty is often associated with unmet demand [43]. A basic assumption of the transportation problem is that the cost of transportation is directly proportional to the number of units transported. However, this assumption cannot be justified in many real world situations, because transportation cost are often cheaper when full truck loads are used.

1.2 Purpose of transportation modelling

The transportation model can be used as a comparative tool providing business decision makers with the information they need to properly balance cost and supply. It involves finding the lowest-cost plan for distributing stock or goods from multiple origins to multiple destinations that demand these goods. This model can be used to compare location alternatives in terms of their impact on the total distribution costs for a system. It is subject to demand satisfaction at markets supply constraints. It also determines how to allocate the supplies available from various factories to the warehouses that stock or demand those goods, in such a way that total shipping cost is minimised.

1.3 The nature of transportation costs

Transportation cost functions may be non-linear because of quantity discounts, or price breaks, etc. Also, a fixed charge may be incurred when at least a unit of commodity is sent from a given source to a given destination. Thus, even if a small quantity is transported on some arcs of the transportation network, a fixed charge must be paid. In practical applications , the fixed charge may represent the cost of renting a vehicle; toll charges on a highway; landing fees at an airport; set-up costs for machines in a manufacturing environment; time to locate a file in a distributed database system, or the cost of building roads, etc. In the presence of such one-time costs, the transportation problem is called the *fixed charge transportation problem* (FCTP).

The FCTP arises not only in distribution, transportation, scheduling, and location systems [2], but also in allocation of launch vehicles to space missions [36], solidwaste management [42], process selection [19], and teacher assignment [21]. In practice, the FCTP is difficult to solve exactly, and although it has attracted considerable research attention in the literature, current state-of-the-art exact solution methods are only able to consistently solve instances with up to 15 sources and 15 sinks [33].

The FCTP differs from the linear (or standard) transportation problem only in the non-linearity of the objective function. When a TP is associated with an additional fixed cost for establishing the facilities of fulfilling the demand of customers, then it is called a fixed charge transportation problem. In the FCTP, a fixed charge is associated with each route that can be opened, in addition to the variable transportation cost proportional to the amount of goods shipped. A FCTP is a special case of the general fixed charge problem. In an FCTP, a single commodity is shipped from origin (source, supply) locations to destination (sink, demand) locations. The FCTP has been a popular research topic in mathematical programming for quite some time. This problem is characterized entirely by the presence of a transportation network structure [37].

1.4 Background on transportation and fixed charge problem

Most operations research scholars are familiar with transportation problems. Every city knows the frustration and delay of congestion and its high cost to the public and to shippers of goods. The consequences of congestion are many. The accident rate rises, traffic moves very slowly – only a few kilometres per hour on some central streets during rush periods. Millions of rands are lost each year by delay to individuals, motors and trucks. Once quiet, residential streets are invaded by the noise, fumes, and danger of traffic; main ways are no longer able to carry the load; and in central areas themselves the pedestrians lose out to the mass of vehicles in the

streets.

The general fixed charge problem (FCP) is one of the more challenging problems of mathematical programming. Its constraint structure (a set of linear equations) is identical with that of a linear programming (LP) problem. The LP problems (including transportation problem) can be solved rapidly by widely available LP software. The existence of the fixed charges in the FCP objective function makes it NP-hard (non-deterministic Polynomial-time hard) and has prevented the development of any extensive theory for its solution. Only medium size problems are solvable using available integer programming (IP) software [22]. The FCP has a wide variety of classic applications that have been documented in scheduling, facility location including capacitated warehouse location problem, portfolio selection, and in fleet routing.

1.5 Problem description

If the notation from the transportation problem is used, the following additional notation is needed to formulate the FCTP mathematically. Let f_{ij} be die fixed cost that is incurred when a positive amount of commodity is transported from source i to destination j. Furthermore, let y_{ij} be a zero/one variable, taking the value one if a positive amount is transported from i to j or else it is zero. In general the objective of the FCTP then becomes to

$$\text{minimize } z = \sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}x_{ij} + f_{ij}y_{ij}) \tag{1.2}$$

$$\text{subject to} \quad \sum_{j=1}^{n} x_{ij} = s_i \qquad i = 1,\ldots,m, \tag{1.3}$$

$$\sum_{i=1}^{m} x_{ij} = d_j \qquad j = 1,\ldots,n, \tag{1.4}$$

$$My_{ij} \geq x_{ij} \qquad \begin{cases} i = 1,\ldots,m \\ j = 1,\ldots,n, \end{cases} \tag{1.5}$$

$$x_{ij} \geq 0 \qquad \begin{cases} i = 1,\ldots,m \\ j = 1,\ldots,n, \end{cases} \tag{1.6}$$

$$y_{ij} \geq 0/1 \qquad \begin{cases} i = 1,\ldots,m \\ j = 1,\ldots,n, \end{cases} \tag{1.7}$$

where M is a large number.

The problem considered of this thesis is to propose, implement and compare heuristic solution approaches to solve the FCTP given in formulation (1.2)–(1.7).

1.6 Aim and objectives

The FCTP is an interesting integer programming problem. The simplicity of the problem statement and difficulty of solution makes this an elegant and hard mathematical programming problem to solve. To an operation research practitioner, the numerous applications in the area of distribution makes practical solution techniques a considerable interest.

1.6.1 Aim of the study

The aim of this study is to determine the best routes to fully satisfy the destination requirements within the operating production capacity constraints at the minimum possible cost.

1.6.2 Objectives of the study

The objectives of the study are:

1. study the literature on the FCTP,
2. develop different classes of heuristics for the FCTP,
3. evaluate the heuristics in terms of computational time and solution quantity,
4. determine good solutions using a primal approach, and
5. determine good solutions using a primal-dual approach.

1.7 Significance of the study

It is well-known that the TP is one of the important traditional optimisation problems, and it has also been widely applied in real-life such as distributing systems, job assignment and transportation. Many researchers have developed solution procedures for various types of FCTPs. The solution procedures are developed mainly on finding the best or rather optimum solutions to the FCTP within a minimal amount of time. This study aims at extending on the knowledge of these solution approaches by suggesting new heuristic solution approach.

1.8 Thesis outline

In this chapter, the scope of the FCTP, various FCTP environments, the advantages and challenges of considering the FCTP are discussed. The rest of the thesis is organised as follows:

- In Chapter 2 of the thesis a brief review of the current literature on heuristic methods and on algorithms for the exact solution approach of FCP are discussed.
- Chapter 3 describes the characteristics, assumption and mathematical formulation for the FCTP in this thesis. Data descriptions is also presented in this chapter.
- In Chapter 4 the attention is directed to the general FCTP. Several methods are compared in determining the initial solution and the one with a better starting solution is considered for improvement. Methods of relaxating FCTPs are also introduced. Results are obtained and their computational cost are indicated.
- In Chapter 5 the computational times of the different solution approaches are presented and discussed.
- In Chapter 6 the thesis concludes with a summary of important results and recommendations for future research. Some speculations are offered as guidelines for future work

CHAPTER 2

Literature survey

Contents

2.1 Exact solution methods 7
2.2 Heuristic solution methods 10
2.3 Chapter summary 13

Transportation problems arise in a large number of production and transportation systems, and they can often be practically modelled as fixed charge transportation problems (FCTP). In the past several decades, many researchers have developed solution procedures for various types of FCTPs. The FCTP continues to be an interesting area of research. This chapter reports the methods/algorithms that are mostly used for solving fixed charge transportation problems. The approached have been separated into two categories, namely *exact approaches* and *heuristic approaches*. These approaches differ in that the exact approaches guarantee optimality whereas the heuristic approaches do not. Exact and heuristic methods for optimisation are sometimes regarded as belonging to entirely different categories. For this thesis, heuristic approaches will be considered.

2.1 Exact solution methods

Exact algorithms are typically deterministic and predictable and are not subject to chance if repeated. Exact methods contain a finite set of instructions to solve a problem and for integer problems generally take the form of branching and bounding or other forms of exhaustive search. Some characteristics of exact approaches includes:

- **finite list**: requires algorithm to have a finite set of actions,
- **convergence criteria**: an algorithm should ultimately converge to an optimal solution and not go on forever, and
- **solution guarantee**: an optimal solution is always guaranteed.

Exact methods may thus be expressed as a finite list of well-defined instructions for calculating a best value for a function [29]. Usually, an exact method is the method of choice as it can solve an optimisation problem to a known optimal solution. Unfortunately, larger problem instances

often become untractable and can hardly be solved using exact methods as the computational effort becomes too much for even powerful computers to handle.

The first formulation of the fixed charge problem was proposed by Hirsch and Dantzig [19] and showed that the optimal solution will lie at an extreme point. Since its initialization, it has been widely applied in many decision making and optimisation problems. Based on this finding, Murty [27] devised the first exact procedure to solve the fixed-charge problem by ranking the extreme points. He then presented only one sample problem which was solved by hand. However, it works effectively only when the problem is non-degenerate and the fixed charges are quite small as compared with the values of the variable cost. Murty also found that it is not necessary to rank all the extreme points since it is possible to determine upper and lower bounds for the problem.

One approach to solving FCTP involves a mixed integer programming formulation. Gray [17] attempted to find an exact solution to the fixed charge transportation problem by decomposing the problem into a master integer program and a series of transportation subproblems. This subproblems involves only continuous variables, and hence can be solved as a linear programs, where there exist a structure of a transportation problems in which only certain routes are to be opened. This method makes use of the upper bound on the fixed charge, particularly useful for problems in which the fixed charge dominate. The approach used is an alternate approach to Murty [27], which searches among the extreme points according to their associated fixed charges. Gray extensively exploited the special structure of the transportation problem to improve computability and noticed that variation between the variable cost and the fixed costs sometimes has an impact on the effectiveness of the algorithm. The algorithm is found to be most efficient when the fixed costs are large compared to the variable costs.

Hultberg and Cardoso [21] formulated a basic model of assigning classes to professors, such that the average number of distinct subjects assigned to each professor is minimized. The problem turned out to be a FCTP which in some cases correspond to finding a basic solution of a transportation problem which is as degenerate as possible. The model was treated as a special case of the pure fixed charge transportation problem in which all the fixed costs are equal to one. However, the special cost structure of this model allows the existence of an alternative formulation of the problem which leads to a more direct approach to its solution. On this formulation it has been proven that the problem is a NP-hard problem. An exact branch-and-bound algorithm based on its alternative formulation was outlined. It did not allow backtracking but instead stopped after finding the first feasible solution. The results are found to be impressive and the algorithm is said to be effective.

Adlakha *et al.* [3] developed an analytical branching method to solve the FCTP. The method starts with a linear formulation of the transportation problem, which converges to an optimal solution by sequentially separating the fixed cost and finding a direction to improve the value of the linear formulation. This method is based on the computation of a lower bound and an upper bound embedded within a branching process. The algorithm converges to an optimal solution as the lower and upper bounds are continually tightened. The optimum solution is achieved when the two bounds are matched. The number of branching stages does not depend on the size of the problem, but on the number of partially loaded cells in the optimal solution. This method is found to be a convergence method, unlike classical branch and bound, which go through all branches with some improvements like excluding some branches, or limiting solution to some areas.

Kowalski *et al.* [24] then tried to improve the approach by Adlakha *et al.* [3] by accelerating the solution. This method solves FCTP by decomposing the problem into series of smaller sub-problems and can be useful to researchers for solving fixed charged problems of any size. A

criteria from Kowalski [22], which has only non-degenerate distributions, has been considered in the test as a benchmark and the solutions was almost equal. This method provides an alternative to solving small scale problems in a way without using computer software. The exact branch-and-bound method is applicable to small problems only, since the effort to solve an FCTP grows exponentially with the size of the problem. Though the branch-and-bound method has many applications, it has been found to be one of the most effective methods to solve the FCTP.

Adlakha *et al.* [4] provided a new approach of approximating and solving FCTP by proposing a novel approximation for the objective function to obtain lower bounds. The lower bound was found to be much superior to the linear bound developed by Balinski [8] and yields optimal solutions to a number of randomly generated problems in an experimental design. They introduced a superior lower bound that work well with other methods. After the adjustment of the variable and fixed costs, computational experiments were carried out to investigate the effectiveness of the proposed approach. The approximations was then found to be easily programmed and implemented for large FCTP using any non-linear problem solver. The proposed method was presented using Balinski's example as a benchmark instance and optimal solutions were attained, although it required more computational time than the linear Balinski approximation.

An exact method for the FCTP has been proposed by Roberti *et al.* [33] based on a new integer programming formulation involving an exponential number of binary variables. Each binary variable corresponds to a feasible supply pattern from sources to sinks. They showed that the lower bound provided by the linear relaxation of the new formulation introduced on their paper is stronger than the optimal solution cost of the linear relaxation problem. Therefore several classes of valid inequalities that are shown to significantly improve this lower bound was proposed. The new formulation was used to develop a column-and-cut generation method to compute a valid lower bound on the FCTP and exact branch-and-price algorithm for solving it. Benchmark instances from Agarwal and Aneja [5] has been used to experiment with the proposed method, in conjunction to their randomly generated instances, and extensive computational results indicated that the new exact method is superior to that of the benchmark.

Sanei *et al.* [35] considered the FCTP under uncertainty, particularly when the direct and fixed costs are the generalized trapezoidal fuzzy numbers. As far as known, with regards to solving the fuzzy fixed-charge transportation problem, no research has been done. Therefore, any method which provides a good solution for it will be distinguished. Firstly, the fuzzy fixed-charge transportation problem was converted into the fuzzy transportation problem by applying the Balinski [8] relaxation. That became a linear version for the fuzzy fixed-charge transportation problem for the next stage, and then, tried to obtain a fuzzy basic feasible solution for the linear version of the fuzzy fixed-charge transportation problem by using one of the well-known transportation methods (i.e. Generalized North-West Corner method, or the Generalized Fuzzy Vogel's approximation method). For the improvement of the solution, the fuzzy modified distribution method was used. An approximation solution for the optimal solution obtained to the fuzzy fixed-charge transportation problem has been found. The proposed method obtained both lower and upper bounds on the fuzzy optimal value of the fuzzy fixed-charge transportation problem which can easily be obtained by using the approximation method.

It is commonly accepted in the literature on the FCTP that exact solution algorithms are not very useful in practice since, except for small dimension problems, the computation time required is usually excessive [37]. The main reason behind this is that the most commonly used relaxations taking part in the branch-and-bound methods are weak for the FCTP. In most of the references cited above to the branch-and-bound algorithms, the most commonly used relaxation is simply the linear relaxation. And this relaxation is not strong for the FCTP, especially for the large-sized problems. To overcome these problem, heuristic optimisation methods are used.

2.2 Heuristic solution methods

Heuristics are a very interesting concept. It has a different character than the exact methods, exploiting local search or an imitation of a natural process [20]. Some problems are hard to solve and one may not be able to get an acceptable solution in an acceptable time. In such cases a good, and not necessarily the best, solution may be calculated much faster, by applying some intelligent choices. This is called a heuristic approach.

Heuristics may not explore all possible states of the problem, or will begin by exploring the most likely ones. In some cases the search is not for the best solution, but for any solution fitting some constraint. A good heuristic would help to find a solution in a shorter time, but may also fail to find any if the only solutions are in the states it chose not to try. Heuristic approaches often have no proof of correctness since it may involve random elements, and may not yield optimal results. In many problems for which no efficient algorithm exist to find an optimal solution, there exist a heuristic approach that can yield near-optimal results in an acceptable time. Given the great computational difficulty of the FCTP, many heuristic methods have been developed over several decades.

Transportation problems are a type of a network problem in which a feasible solution has a spanning tree topology. Thus, a spanning tree-based representation would be appropriate for the problem. Hajiaghaei-Keshteli *et al.* [18] presented several ideas to handle the spanning tree-based genetic algorithm for the FCTP. A genetic algorithm based on spanning trees was considered and a pioneer method was presented to design a chromosome that does not need a repairing procedure for feasibility. Six crossover and four mutation operators were developed for this problem from the genetic algorithm literature, of which a mutation operator was presented considering prüfer number representation. The Taguchi parameter design method was employed to adjust the parameters and operators of the proposed genetic algorithm. They found that the robustness of the algorithm may be improved by fine-tuning the genetic algorithm parameters and operators, relating the population size, reproduction percentage, mutation probability, cross over and mutation types.

Transportation of goods in a supply chain from plants to customers can also be modelled as a two-stage distribution problem. Ray and Rajendran [31] developed a genetic algorithms as a two-stage transportation with two scenarios. The first scenario considers the per-unit transportation cost and the fixed cost associated with a route, coupled with unlimited capacity at every distribution centre. The second scenario considers the opening cost of a distribution centre, per-unit transportation cost from a given plant to a given distribution centre and the per-unit transportation cost from the distribution centre to a customer. An attempt was made to represent the two-stage fixed charge transportation problem as a single-stage FCTP and solve the resulting problem using the genetic algorithm. Benchmark problem instances from the literature were used to test the performance of the proposed method and the best existing methods algorithms for the two scenarios. The proposed algorithm yielded better solutions than the respective best existing algorithms for both scenarios.

Lotfi and Moghaddam [25] developed a new genetic algorithm to find a heuristic solution for the FCTP, considering the objective function in a linear or non-linear term. A new genetic algorithm was developed in order to find the best heuristic solution after providing a comparative analysis for several possible representation methods for transportation problems (i.e. matrix-based, direct transportation tree, basic feasible solution, spanning tree-based, priority-based). A novel priority-based genetic algorithm was proposed for such NP-Hard problems, in which the relevant procedure was developed and three new operators were designed. A priority-based decoding procedure proposed by Gen and Altiparmak [15] was modified to adapt with the FCTP

structure. That made their proposed method to be applicable for relatively large-sized problems. Two famous benchmark instances from the literature was used for testing the performance of the priority-based genetic algorithm. It was observed that the proposed priority-based genetic algorithm always gives a better solution quality than those available in literature for the FCTP. Thus, according to the achieved results, the priority-based genetic algorithm has the explicit excellence in proportion to the spanning-tree genetic algorithm both in terms of the solution quality and computational time, more especially on medium and large sized problems.

The nonlinear fixed charge transportation problem is a variant of the fixed charge transportation problem. Xie and Jia [44] developed an efficient method to solve nonlinear fixed charge transportation problems, which was formulated using a mixed integer programming model. A minimum cost flow-based genetic algorithm, which can also be called a hybrid genetic algorithm, was employed to solve the nonlinear FCTP. This algorithm was developed based on a steady-state genetic algorithm as framework and minimum cost flow genetic algorithm as decoder. The goal was to develop an efficient method to solve the nonlinear FCTP by means of taking advantage of nonlinear structure and special network structure of the nonlinear FCTP. Two previously addressed problems from the literature with different sizes, were used to evaluate the performance of the proposed algorithm. The hybrid genetic algorithm has found a better near-optimal solution with total cost decreasing exponentially, at the cost of acceptable time. Thus the proposed hybrid genetic algorithm was found to be an efficient and robust method to solve nonlinear FCTP, especially applicable to large scale problems.

Sun *et al.* [37] developed a tabu search approach for the FCTP using recency based and fixed frequency based memories. Two strategies for each of the intermediate and long term memory process were also used, making use of a network based implementation of the simplex method as a local search method. Some randomly generated problems of different sizes and of different ranges of magnitude of fixed costs relative to variable cost were used to evaluate their proposed approach computationally. A comparison of the proposed method with two leading methods previously proposed, one in the category of exact methods and the other in the category of heuristic methods, was done. This showed that the proposed approach obtained optimal or near-optimal solutions more than a thousand times faster than the exact solution algorithm for simple problems. It also dominated the exact algorithm with computational time for more complex problems. In comparison with the heuristic approach, the proposed procedure required about the same amount of solution time with solutions at least as good. However, for larger problems and for problems with higher fixed relative to variable costs, the tabu search procedure was $3-4$ times faster than the competing heuristic, and found significantly better solutions in all cases.

The iterated local search framework has established itself as one of the most effective metaheuristic approaches for finding approximate solutions of hard combinatorial optimisation problems. Buson *et al.* [11] proposed an iterated local search heuristic based on the utilization of reduced costs for guiding the restart phase. The reduced costs were obtained by applying the lower bounding procedure, that computes a sequence of non-decreasing lower bounds by solving a three-index mathematical formulation of the problem. The focus was on exploiting dual bounds to guide the perturbation phase. The perturbation phase is a purely random procedure that generates a solution from another solution. Two sets of benchmark instances from the literature were considered in testing the performance of the proposed method. The first set was used to evaluate the state-of-the-art heuristics for the problem, where the proposed heuristic was able to provide new best-known upper bounds on all tested open instances. On the second set of instances, which was recently introduced for testing the currently best exact method for the problem, the new heuristic was able to provide certainly good upper bounds within short com-

puting times. The proposed heuristic was tested and found optimal solutions for some of the instances for which the optimal cost is known. On all open instances, the proposed method has improved known solutions, within a few minutes of computing time.

Several genetic algorithms based on spanning tree and Pŕufer number were presented in an attempt to solve fixed charge transportation problems, and most of such methods do not guarantee the feasibility of all chromosomes generated. Altassan *et al.* [7] introduces an artificial immune system for solving the FCTP using the genetic algorithm that is flexible enough to solve both balanced and unbalanced FCTP without introducing any dummy supplier nor a dummy customer. Instead of using the spanning tree and Prüfer numbers, a coding schema is designed and algorithms are developed for coding such schema and allocating the transported units. Some mutation functions were developed and their performance compared to select the best one. The repairing procedures were not necessary since all the generated antibodies were feasible. The performance of this algorithm and its solution quality prove that the artificial immune system for solving the FCTP is highly comparative and can be considered as a viable alternative to solve the FCTP.

Aguado [6] developed a new heuristic approach for FCTP by combining the Lagrangean relaxation, Branch-and-Bound and heuristics in different phases. The method is basically based on the solution of the sub-problems, that contains only a subset of all the variables. Reducing the size of the original problem was mainly to make it easier for the problems to be solved. This algorithm consists of three phases whereby on the first phase, either the Lagrangian relaxation or the Lagrangian decomposition is applied to obtain both a lower bound and the Lagrangian reduced cost of all variables. At this stage the problem is still too difficult to solve, so no attempt is made to obtain good solutions. On the second phase, from the previously computed reduced costs, one or several sub-problems, with the same structure as the original problem but with fewer variables are selected. The langrangean decomposition is applied again to each sub-problem and the best heuristic solution is attained in this phase. On the last phase, enumeration is resorted by applying a standard branch and cut algorithm to the sub-problem that produced the best solution in phase two. Thus improving the final solution. In order to study the effectiveness of the method, they used the same comprehensive FCTP test instances as Sun *et al.* [37] and Glover *et al.* [16] where the tabu search method and the parametric ghost image process method, respectively, has been applied. The proposed method could obtain similar or better solutions than the two state-of-art algorithm for problems with lower ratios.

The more-for-less paradox in a standard transportation problem and FCTP occurs when it is possible to ship more total goods for less (or equal) total cost, while shipping the same amount or more from each origin and to each destination and keeping all the shipping cost non-negative. Adlakha and Kowalski [1] developed an efficient procedure for obtaining a more for less solution of a fixed-charge transportation problem by simply locating the absolute points of its relaxed transportation problem. The procedure looks for cells that would always be used in any optimal solution due to cost efficiency. The absolute points reduce the dimensions of the cost matrix of the relaxed transportation problem and consequently of the corresponding FCTP. The existing literature does not provide any methods for achieving a more for less solution for an FCTP. Thus, the absolute points provides the candidate locations for the more for less solution in the FCTP. The identification of an absolute point could also be used as an alternative approximation algorithm for solving FCTPs. It has also been tested that a relaxed transportation problem may not initially have any absolute points when it has an alternate optimal solution.

A simple efficient heuristic procedure for solving the step fixed charge transportation problems was developed by Kowalski and Lev [23]. The step fixed charge transportation problems is a variation of the FCTP where the fixed cost is incurred by activating a route and it includes

a step function. Each time a route is opened or closed, the objective function jumps a step. The unavailability of an algorithm for the step fixed charge transportation problems makes any heuristic method that provides a good solution to be considered useful. The approach to this method does not differ to the FCTP where at first a good initial solution is obtained and converge on a better solution afterwards. Two heuristic algorithms which are similar to the Balinski [8] method are proposed to determine a good initial solution. Several new aspects of the problem showing the differences and the similarities to the fixed-charge problems was introduced.

The primary reason why many researchers resort to a heuristic or a meta-heuristic approach is due to the long time taken by exact algorithms. This is true when the problem is complex and the search space for the solution is very large and grows exponentially [29]. Usually in academic applications, it is often required to arrive at the true optimal solution. However, in practice there always exist a concept of acceptable solutions. This is due to the valuation of time taken by the computer or an individual to arrive at that solution.

2.3 Chapter summary

Based on the literature, branch and bound algorithms are extensively proposed to solve the FCTP, but still is a very computational intensive procedure for solving large problems. Since exact methods guarantees optimal solutions, it is only efficient to be applied on small fixed charge transportation problems. Relative to the transportation problem, the FCTP is more difficult to solve due to the presence fixed costs which causes discontinuities in the objective function [30].

The popularity of heuristic methods continue to exist because exact methods cannot always be applied due to the complexity of nonlinear FCTP. Some heuristic approaches were attempted in solving the FCTPs of any size, although optimal solutions are not guaranteed. Most authors used randomly generated benchmark instances to compare their solutions with those in the literature. Several algorithms' performance were improved and better solutions were obtained than those found in literature. Thus, there is always a room for improvement in most heuristics.

CHAPTER 3

Data

Contents

3.1 Description of the benchmark instances 15
3.2 Linear formulation 17
3.3 Implementation 18
3.4 Chapter summary 18

It is difficult to compare the performance of algorithms simply by looking at their specifications (features). In order to evaluate the performance of the tabu search (TS) algorithm developed in this thesis for solving the FCTP, secondary data from the literature is used. The data required for the problem include the number of plants and customers. In the absence of real data, to test the proposed algorithm, three levels of data are considered.

3.1 Description of the benchmark instances

The first level of data contains Dataset 1 (i.e. data introduced by Agarwal and Aneja [5]) and Dataset 2 (i.e. data introduced by Roberti *et al.* [33]). The second level of data is a set of subgroup of similar instances within a dataset with the same parameter settings. Therefore, 30 instances on a set of 15 origin and 15 destinations with $B = 20$ are considered. The objectives of this experiments was to examine the effect of the magnitude of the flow costs c_{ij} versus the fixed costs f_{ij} on the results. The contribution of flow cost toward the total cost depends not only on c_{ij} values but also on total traffic volume D. Assuming that a feasible solution has $(m+n-1)$ open routes, average traffic per route would be $D/(m+n-1)$, which translates into a flow cost of $c_{ij} \times D/(m+n-1)$ against the fixed cost f_{ij} for that route. Define $\theta = [c_{ij} \times D/(m+n-1)]/f_{ij}$ such that the cost contributions of flow costs and fixed costs are expected to be roughly in the ratio of $\theta : 1$. Three scenarios are examined with $\theta = 0.0,\ 0.2,$ and 0.5, respectively, and scale the c_{ij} values appropriately for the desired value of θ.

Dataset 1 shall be referred to as the small sized instances and their quantities, s_i and d_j, were randomly generated in the interval [1,20] with uniform distribution. Fixed and variable costs were generated in the interval [200, 800], but unit costs were scaled to maintain a predefined ratio θ between the total variable and fixed costs. The instances are grouped into three classes characterized by different values of the parameter θ (i.e. $\theta = 0.0,\ 0.2$ and 0.5). Table 3.1 gives the detailed summary about the problem parameters for these problems. For ease of understanding, the following terminologies will be used to describe the datasets:

Level 1: This contains a whole dataset. Thus, dataset generated by Agarwal and Aneja [5] (this shall be called Dataset 1) and dataset generated by Roberti *et al.* [33] (this shall be called Dataset 2).
Level 2: This level contains subgroups of similar instances within a dataset (i.e. Dataset 1 and Dataset 2). The groups of instances in a dataset have the same parameter settings. Each set of instance has a name, labelled based on the author. For example, the third set in Agarwal and Aneja will be reffered to as Set A3 and the seventh set of Roberti *et. al.* will be Set R7.
Level 3: This instance contains one FCTP.

Set	Size	B	θ	**Range of c_{ij}**	Range of f_{ij}	Range of a_i	Range of b_i
A1	15×15	20	0.0	**[200,800]**	[200,800]	[20,50]	[20,50]
A2	15×15	20	0.2	**[200,800]**	[200,800]	[20,50]	[20,50]
A3	15×15	20	0.5	**[200,800]**	[200,800]	[20,50]	[20,50]

TABLE 3.1: *Summary of Dataset 1 test problems*

Dataset 2 contains 180 instances with up to 70 origins and 70 destinations. These instances also feature s_i and d_j values ranging in the interval [1,B], B∈{20,50}; $\theta \in \{0.0, 0.2, 0.5\}$; and fixed costs as $c_{ij} = [(\theta f_{ij}(m+n-1))/(\sum_{i \in S} a_i)]$. The instances are grouped into six sets of 30 instances characterized by different values of parameter B and θ. Table 3.2 gives the detailed summary about the problem parameters for these problems.

Set	Size	B	θ	**Range of c_{ij}**	Range of f_{ij}	Range of a_i	Range of b_i
R1	30×30	20	0.0	**[200,800]**	[200,800]	[20,50]	[20,50]
R2	50×50	20	0.0	**[200,800]**	[200,800]	[20,50]	[20,50]
R3	70×70	20	0.0	**[200,800]**	[200,800]	[20,50]	[20,50]
R4	30×30	20	0.2	**[200,800]**	[200,800]	[20,50]	[20,50]
R5	50×50	20	0.2	**[200,800]**	[200,800]	[20,50]	[20,50]
R6	70×70	20	0.2	**[200,800]**	[200,800]	[20,50]	[20,50]
R7	30×30	20	0.5	**[200,800]**	[200,800]	[20,50]	[20,50]
R8	50×50	20	0.5	**[200,800]**	[200,800]	[20,50]	[20,50]
R9	70×70	20	0.5	**[200,800]**	[200,800]	[20,50]	[20,50]
R10	20×20	50	0.0	**[200,800]**	[200,800]	[20,50]	[20,50]
R11	30×30	50	0.0	**[200,800]**	[200,800]	[20,50]	[20,50]
R12	40×40	50	0.0	**[200,800]**	[200,800]	[20,50]	[20,50]
R13	20×20	50	0.2	**[200,800]**	[200,800]	[20,50]	[20,50]
R14	30×30	50	0.2	**[200,800]**	[200,800]	[20,50]	[20,50]
R15	40×40	50	0.2	**[200,800]**	[200,800]	[20,50]	[20,50]
R16	20×20	50	0.5	**[200,800]**	[200,800]	[20,50]	[20,50]
R17	30×30	50	0.5	**[200,800]**	[200,800]	[20,50]	[20,50]
R18	40×40	50	0.5	**[200,800]**	[200,800]	[20,50]	[20,50]

TABLE 3.2: *Summary of Dataset 2 test problems*

For each possible combination of θ and B, each subgroup contains 30 instances of different sizes (10 instances for each size). When $B = 20$, the size of instances can be 30×30, 50×50, and 70×70; whereas when $B = 50$, the size can be 20×20, 30×30, and 40×40. Therefore, this set of instances is made up of 18 subgroups, where each subgroup contains 10 instances and is characterized by

the size and the values of θ and B. All instances of Datset 1 and Dataset 2 fulfil the assumption made in Section 1.1 that $\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j$. The dataset instances were kindly provided by the author, Roberti.

3.2 Linear formulation

The FCTP is significantly harder to solve because of the discontinuity in the objective function, Z, introduced as the fixed cost. Balinski [8] has provided a heuristic solution for FCTP. Assuming the fixed cost as f_{ij}, the Balinski matrix is obtained by formulating a linear version of FCTP by relaxing the integer restriction on y_{ij}, with the property that

$$y_{ij} = \frac{x_{ij}}{M_{ij}}, \tag{3.1}$$

where

$$M_{ij} = \min\ (s_i, d_j). \tag{3.2}$$

The relaxed transportation problem (RTP) of the FCTP would then simply be a standard TP with unit transportation costs of shipping through the route (i, j) as follows

$$C_{ij} = c_{ij} + \frac{f_{ij}}{M_{ij}}, \tag{3.3}$$

where f_{ij} is a fixed cost and RTP may then be written as to

$$\begin{aligned} \text{minimise } Z^* &= \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}x_{ij} \\ \text{subject to } \sum_{i=1}^{n} x_{ij} &= s_i \qquad i = 1, \ldots, n, \\ \sum_{j=1}^{m} x_{ij} &= d_j \qquad j = 1, \ldots, m, \\ x_{ij} &\geq 0 \qquad i = 1, \ldots, n; \quad j = 1, \ldots, m. \end{aligned} \tag{3.4}$$

The optimal solution X'_{ij} to the RTP problem can easily be modified into a feasible solution of X', y' of FCTP as follows:

$$y'_{ij} = \begin{cases} 0, & \text{if } X'_{ij} = 0, \\ 1, & \text{if } X'_{ij} > 0. \end{cases} \tag{3.5}$$

Balinski also shows that the optimal value X'_{ij} of RTP provides a lower bound to the optimal value Z^* of FCTP and modified feasible solution $\{X', y'\}$ provides an upper bound

$$\sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}X'_{ij} \leq Z^* \leq \sum_{i=1}^{n}\sum_{j=1}^{m}(c_{ij}X'_{ij} + f_{ij}y_{ij}). \tag{3.6}$$

For every loaded location (i, j) the cost of the fixed charge function formulation is greater than the corresponding cost of the relaxed integer restriction function (Balinski approximation). The Balinski linear approximation can be represented as in Figure 3.1.

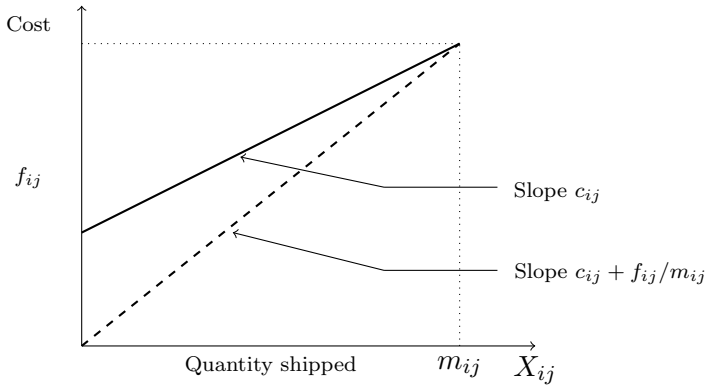


FIGURE 3.1: *Shipping cost as a function of quantity shipped along route* (i, j) *for FCTP.*

3.3 Implementation

Algorithms were coded to solve the data instances reported in this chapter. All of the algorithmic features were implemented and coded in Python (version 3.3.3)[40] and has been run on Intel(R) Core(TM) i7-3770 CPU @3.40GHz with 8.00GB of RAM and 64-bit Operating System. It was run on a DELL computer running on Windows 10 Enterprise 2015 LTSB. The program is designed to solve problems whose product, m, n, is of any dimensions – but the problem must be a balanced transportation problem (i.e. $\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j$). The code terminates after k iterations, which the user must input at the beginning of the running process.

3.4 Chapter summary

Benchmark instances from the literature has been described in this chapter. The structure of the datasets, the interpretations of their parameters and the linear formulation for the Balinski transformation have been discussed. Lastly, the specifications of the computer used to run the algorithms is explained.

CHAPTER 4

Model

Contents

4.1 Mathematical model and descriptions 20
4.1.1 Existence of feasible solution *23*
4.1.2 Basic feasible solution *23*
4.2 Tabu search algorithm 24
4.3 The initial solution 26
4.3.1 Heuristics methods of finding initial solutions *26*
4.3.2 Computational results of initial solution methods *29*
4.4 Solution improvement 31
4.4.1 Stepping stone method *31*
4.4.2 Selection criteria *33*
4.5 Tabu list 35
4.6 Aspiration criteria 35
4.7 A primal-dual method for solving transportation problems 36
4.8 Chapter summary 39

This chapter reviews the proposed solution methodology and approach for handling fixed charge transportation problem (FCTP) in this thesis. The FCTP seeks to minimise the total shipping costs of transporting goods from m origins (each with a supply s_i) to n destinations (each with a demand d_j), when the unit shipping cost and fixed cost from an origin, i, to a destination, j, is c_{ij} and f_{ij}, respectively. The fixed charge transportation problem (FCTP) differs from the linear transportation problem (TP) only on the nonlinearity of the objective function [32]. While not being linear in each of the variables, the objective function of the FCTP has a fixed cost associated with each variable. The fixed cost for each variable is incurred when and only when that variable is at a positive level in the solution.

Since the fixed charge problem was initialised by Hirsch and Dantzig [19], it has been widely applied in many decision making and optimisation problems. A brief description of the solution procedure for the FCTP is presented in Figure 4.1. In any given fixed charge problem, the transportation matrix is formulated. Before attempting to solve the problem, it must be verified if it is a balanced transportation problem. Should the problem be unbalanced, a dummy source (destination) is introduced with a zero variable cost and zero fixed cost in each route and a supply (demand) taking a difference of $\sum s_i$ and $\sum d_j$.

The next step is to determine the initial feasible solution, which should satisfy the $n+m-1$ rule. The $m+n-1$ rule is when a fixed charge transportation solution has $n+m-1$ positive cells,

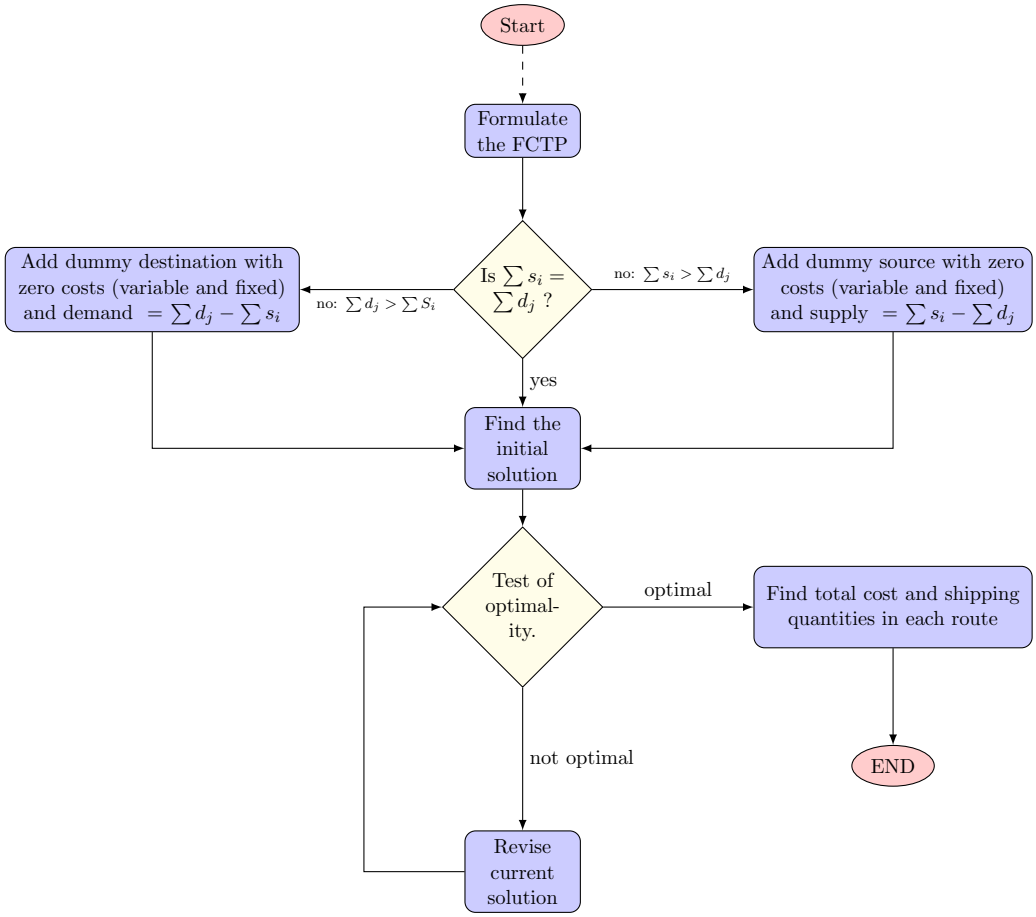


FIGURE 4.1: *Flow chart of the fixed charge transportation problem approach*

where n is the number of supply points and m is the number of demands destination. Different criteria of determining the initial solution exist and a good initial solution can sometimes be the best or an optimal solution. Again, there exist several methods of improving the initial solution of a FCTP. The solution is then evaluated and improved using any optimisation method until an optimal or near optimal solution is obtained. The total shipping costs and the corresponding shipping quantities can then be computed to the optimal solution. This solution presents the routes, with the corresponding shipping quantities, that are to be opened to ship goods from the supply points to the demand destination at a minimal cost.

4.1 Mathematical model and descriptions

The variable cost can be associated with linear or nonlinear variables, and therefore the objective function will be linear or nonlinear. Having the key value in real-world applications, such as electric power transport systems, the nonlinear FCTP has recently attracted a number of researchers. The objective of this formulation is to minimise the cost incurred in transporting the goods from the suppliers to the customers considering the possible combination of routes.

As an example, suppose a company has m warehouses and n retail outlets. A single product is to be shipped from the warehouses to the outlets. Each warehouse has a given level of supply, and each outlet has a given level of demand. The transportation cost between every pair of warehouse and outlet is given, and these costs are assumed to be linear. More explicitly, the

assumptions are:

- the demand for a single-product is considered,
- the number of suppliers and the corresponding capacities are known,
- the number of customers and the corresponding demands are known,
- the sum of demands and the sum of supplies are equal (balanced transportation problem),
- customers can be supplied with products from more than one supplier, and
- transportation damages or losses are not considered.

The variables in this model of the FCTP will hold the values for the number of units shipped from one origin to a destination. The following input parameters are considered in our formulation. Let

n	be number of origins,
m	be number of destinations,
i	be index for origins, $(i = 1, 2, \ldots, n)$,
j	be index for destinations, $(j = 1, 2, \ldots, m)$,
a_i	be amount of supply at origin i,
b_j	be amount of demand at destination j,
c_{ij}	be variable transportation cost from origin i to destination j, forming the matrix C and
f_{ij}	be fixed transportation cost associated with route (i, j), forming the matrix F.

The fixed-charge transportation problem is traditionally formulated as a mixed 0-1 integer programming problem with the objective to

$$\begin{aligned} \text{minimise } Z &= \sum_{i=1}^{n}\sum_{j=1}^{m}(c_{ij}x_{ij} + f_{ij}y_{ij}) \\ \text{subject to } &\sum_{i=1}^{n} x_{ij} = s_i \qquad i = 1, \ldots, n, \\ &\sum_{j=1}^{m} x_{ij} = d_j \qquad j = 1, \ldots, m, \\ &x_{ij} \geq 0 \qquad i = 1, \ldots, n; \quad j = 1, \ldots, m, \end{aligned} \tag{4.1}$$

$$y_{ij} = \begin{cases} 0, & \text{if } x_{ij} = 0, \\ 1, & \text{if } x_{ij} > 0, \end{cases}$$

where the nonnegative variable x_{ij} represents the quantity of units shipped from source i to destination j and a binary variable y_{ij} taking a value 1 if and only if x_{ij} is positive (i.e. arc

(i, j) is used) or 0 otherwise. The FCTP is a variant of the standard TP, which arises when both variable and fixed costs are present. Without loss of generality, assume that

$$\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j, \text{ where } s_i, d_j, c_{ij}, f_{ij} \geq 0. \tag{4.2}$$

Definitions

A feasible solution to a FCTP is a set of non-negative allocations, x_{ij}, that satisfies the row (column) constraints. A feasible solution is said to be a *basic feasible solution* if it contains not more than $n + m - 1$ independent (i.e. not in a loop) non-negative allocations where n is the number of rows and m is the number of columns in a fixed charge transportation matrix. A feasible solution (not necessarily basic) is said to be optimal if it minimises the transportation cost. The basic feasible solutions that contain less than $n + m - 1$ non-negative allocations are called *degenerate* basic feasible solutions. Any basic feasible solution to a FCTP is said to be non-degenerate basic feasible solution if it contains exactly $n + m - 1$ non-negative allocations in independent positions.

Degeneracy

In transportation problems degeneracy exists when the number of cells with non-negative entries is less than the number of rows plus the number of columns minus one $(m + n - 1)$ as shown in Table 4.1. The values on the inner box on each cell represents the unit cost of transporting shipment from source (supply) i to destination (demand) j. Degeneracy may be observed either during the initial allocation when the first entry in a row or column satisfies both the row and column requirements or during the stepping stone method (to be discussed later in this chapter) application, when the added and subtracted values are equal. Degeneracy requires some adjustment in the matrix to evaluate the solution achieved. The form of this adjustment involves inserting some value in an empty cell so a closed path can be developed to evaluate other empty cells. This value may be thought of as an infinitely small amount, having no direct bearing on the cost of the solution.

	A	B	C	Supply
1	12 **3**	10	6 ϵ	3
2	4 **1**	15 **2**	3	3
3	9	7	2 **4**	4
Demand	4	2	4	10

TABLE 4.1: *Transportation tableau showing a degenerate solution*

Procedurally, the value (often denoted by the Greek letter epsilon, ϵ) is used in exactly the same manner as a real number except that it may initially be placed in any empty cell, even though row and column requirements have been met by real numbers. In this project a zero

coefficient is assigned in place of epsilon ϵ, but read it as a nonzero value to allow the creation of a loop. Thus, zero values in a matrix may not be read similarly.

4.1.1 Existence of feasible solution

It can be shown that a balanced supply and demand is a necessary and sufficient condition for the existence of a feasible solution [12]

Theorem 1 *A necessary and sufficient condition for the existence of a feasible solution to a fixed charge transportation problem is that*

$$\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j. \tag{4.3}$$

Proof: The condition is necessary. Suppose there exist a feasible solution to the FCTP. Then,

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} = \sum_{i=1}^{n} s_i \text{ and } \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} = \sum_{j=1}^{m} d_j \tag{4.4}$$

$\Leftrightarrow$

$$\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j \tag{4.5}$$

Hence the necessary part. The condition is sufficient. Let us assume that $\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j = k$(say). If $\lambda \neq 0$ be any real number such that $x_{ij} = \lambda_i d_j$ for all i and j, λ_i is given by

$$\sum_{j=1}^{m} x_{ij} = \sum_{j=1}^{m} \lambda_i d_j = \lambda_i \sum_{j=1}^{m} d_j = k\lambda_i \tag{4.6}$$

or,

$$\lambda_i = \frac{1}{k}\sum_{j=1}^{m} x_{ij} = \frac{s_i}{k} \tag{4.7}$$

Thus

$$x_{ij} = \lambda_i d_j = \frac{s_i d_j}{k}, \text{ for all } i \text{ and } j \tag{4.8}$$

Since $s_i > 0, d_j > 0$, for all i and $j, x_{ij} \geq 0$. Hence, a feasible solution exists.

4.1.2 Basic feasible solution

Cooper and Steinberg [12] also proved that a basic feasible solution for a $n \times m$ FCTP will contain $n + m - 1$ basic variables.

Theorem 2 *The number of basic variables in an $n \times m$ fixed charge transportation tables are $n + m - 1$.*

Proof: Consider an $n \times m$ FCTP with $n-$sources and $m-$destinations. According to the theorem in 4.1.1, out of $n + m$ constraint equations, any one of equations is redundant and it can be eliminated. So, the remaining $n + m - 1$ form a linear independent set.

To proof this, first add n row equations and subtract, from the sum, the first $m - 1$ column equations (4.1), thereby getting the last column's equation. That is

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} - \sum_{j=1}^{m-1}\sum_{i=1}^{n} x_{ij} = \sum_{i=1}^{n} s_i - \sum_{j=1}^{m-1} d_j. \tag{4.9}$$

This can be rewritten as

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} - \left(\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij} - \sum_{i=1}^{n} x_{im}\right) = \sum_{i=1}^{n} s_i - \left(\sum_{j=1}^{m} d_j - d_m\right). \tag{4.10}$$

It thus follows that

$$\sum_{i=1}^{n} x_{im} = d_n \tag{4.11}$$

since

$$\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j. \tag{4.12}$$

This implies that out of $m + n$ constraint equations, only $n + m - 1$ equations are linearly independent. Hence, a basic feasible solution will consist of at most $n+m-1$ positive variables, others being zero. In the degenerate case, some of the basic variables might be zero too. By the fundamental theorem of linear programming, one of the basic solutions will be the optimal solution [28].

Remarks:
(i) When the total capacity equals the total requirement, the problem is called a balanced transportation problem.
(ii) The allocated cells in the fixed charge transportation table are called occupied cells and empty cells are referred to as non-occupied cells.

A metaheuristic algorithm was developed to solve the fixed charge transportation problem, as the major contribution of this thesis. The metaheuristic algorithm combines the principle of greedy search, tabu search algorithm and the stepping stone method.

4.2 Tabu search algorithm

Tabu search (TS) is a metaheuristic method for solving combinatorial optimisation problems. This algorithm was first proposed by Glover in 1986 [14], although it borrowed many ideas suggested before during the 1960s. In the 1990s, the tabu search algorithm became a popular algorithm in solving optimisation problems in an approximate manner and it is now one of the most widespread metaheuristic algorithms [39]. The majority of the research on tabu search use a very restricted domain of the principles of the technique. They were often limited to a tabu list and an elementary aspiration condition. It has successfully been used for tackling different

kind of real-life problems for academic literature such as travelling salesman problem, knapsack problem, and other NP-Hard problems.

It uses flexible memory structures to maintain knowledge about a selective history of the solutions encountered during the search. Tabu search methods guide next searching attempts to move away from local optimal solutions. By giving recently or frequently (or infrequently) visited solutions a tabu status to discourage their selection in the search process [37]. Tabu search has three major components, a *short term* memory process, an *intermediate memory* process, and a *long term* memory process [39], which are described below.

- *The short term memory* is based on a set of tabu conditions and aspiration criteria. Through recency or frequency based memories, tabu search defines a subset of potential moves as tabu or forbidden to avoid repetition.
- *The intermediate memory* is implemented by restricting the search within a set of potentially prosperous solutions to intensify the search.
- *The long term memory* is invoked periodically to lead the search to new regions that might not have been explored as it diversifies the search.

Tabu search manages a memory of the solutions or moves recently applied, which is called the tabu list. This tabu list constitutes the short-term memory. In this study, the four major elements of the tabu search algorithm are 1) find an initial solution, 2) improve the current solution, 3) maintain a tabu list, and 4) stopping criteria. This elements are briefly described in Algorithm 1.

Algorithm 1: Tabu search algorithm

Input : Supplies $\underline{s}$, Demands $\underline{d}$, Variable cost matrix C, Fixed cost matrix F, Balinski matrix γ, and TabuListSize l.
Output: Best solution found.

```
TabuList ← ∅;
Determine initial solution;
tempSol ← greedy(γ);
initialCost ← ObjValCal(tempSol,s,d,C,F);
bestCost ← initialCost;
Improving the current solution;
while stopping criterion is not met do
    currentSolution ← SteppingStone(tempSol);
    newCost ← ObjValCal(tempSol);
    if currentSolution ∉ TabuList then
        if newCost < bestCost then
            bestSolution ← currentSolution;
            bestCost ← newCost;
            tempSol ← bestSolution
        end
        if the length of TabuList < TabulistSize then
            TabuList ← currentSolution;
        else
            delete the first entry in the TabuList;
            TabuList ← currentSolution;
        end
    end
end
return: bestSolution
```

Metaheuristic methods unlike the heuristic methods allow, under certain conditions, the transition to a solution of inferior value of the objective function [38]. These consist of the two main parts. The first part is finding the initialisation solution and the calculation of the objective function. The second part is the process of improving the value of objective function in order to search for an improved solution.

4.3 The initial solution

Any tabu search algorithm requires an initial solution from which all search process begins. Most of the studies that apply tabu search, pay little attention to the initial solution [9]. Generally, finding the initial solution is not that difficult, such as assigning each supplier to a route, and can be obtained with a fast greedy heuristic. The initial solution might have an influence on the quality of the final solution, leaving less of the improvement effort to the tabu search algorithm.

4.3.1 Heuristics methods of finding initial solutions

When initial solutions for a metaheuristic are created, many different approaches can be used. However, good initial solution that will give an important contribution to enhance the final solution can be determined quickly using fast heuristic approaches. Different heuristic methods of determining the initial solutions are considered. In some of the methods, the FCTP is transformed into a classic TP by applying the Balinski transformation described in Section 3.2. This transformation trade-off the variable costs and the fixed costs into a single unit cost (as shown in Algorithm 1). Four heuristic methods are introduced in this thesis from the simplest to the most advanced.

Random allocation method

This method requires less computational time since less computations are involved. In the random allocation method (RAM), the variable costs and the fixed costs are considered by performing a linear formulation. A random cell is chosen from the computed relaxed costs (Balinski matrix) and allocated the maximum possible shipment m_{ij}. The corresponding row (column) is then crossed out if it has been satisfied or adjusted based on the current allocation to the cell. The Balinski matrix is computed as in the random minimum cost method. Then the new allocation is randomly performed, avoiding cells that has already been allocated. This process is repeated until all supplies and demands are satisfied. When exactly one row or column is left, all the remaining cells are basic and are assigned the only feasible allocation. Algorithm 2 presents the steps on applying the RAM method. If all supplies and demands are satisfied, an initial solution is obtained.

Relaxed minimum cost method

The initial feasible solution is obtained using the heuristic approach by employing the standard linear network programming relation. Unlike the classic transportation methods which considers the variable costs, the relaxed minimum cost method (ReMCM) consider both the variable and fixed costs transformed into a unit cost and it gives, on average, a better starting solution (initial basic feasible solution).

Algorithm 2: Random assignment algorithm

Input : Variable costs C, Supplies $\underline{s}$ and demands $\underline{d}$.
Output: Initial basic feasible solution (bfs).

$\gamma, \lambda \leftarrow n \times m$ empty matrix ;
for *all empty cells in* γ **do**
    // Determine the Balinski matrix
    $\gamma_{ij} \leftarrow c_{ij} + f_{ij}/m_{ij}$;
end
while *all supplies* $\underline{s}$ *and demands* $\underline{d}$ *are not satisfied* **do**
    $k \leftarrow$ a random number less than $n \times m$;
    Randomly select cell(i, j) with s_i or $d_j > 0$;
    $\Delta = s_i - d_j$; // difference of supply i and demand j
    if $\Delta \geq 0$ **then**
        // open route ij and assign possible maximum flow to it
        $\lambda_{ij} = d_i$;
        $s_i = s_i - d_j$; // adjust supply i and demand j
        $d_j = 0$;
    else
        $\lambda_{ij} = s_j$;
        $d_i = d_i - s_i$; // adjust supply i and demand j
        $s_i = 0$;
    end
end
return: λ

Algorithm 3 presents the process on how the initial feasible solution is generated using the minimum cost method when Balinski's relaxation approach is considered. The algorithm starts by computing the Balinski matrix and search for a cell with a minimum unit cost. It then allocate the maximum possible quantity (i.e. $m_{ij} = \min\{s_i, d_j\}$) to the cell and recompute the Balinski matrix. The difference between the current and the previous Balinski matrix will lie on the row and column corresponding the cell that had just been allocated (the route that had just been opened) since the supply and the demand corresponding to that cell has been adjusted. This process is repeated until the supply and demand constraints are satisfied. Once the supply and demand constrains are satisfied, an initial solution is obtained and the algorithm terminates.

Random minimum cost method

This method does not differ from the ReMCM except that there is randomness in selecting which route to open in every iteration. The process of cell selection is presented in Algorithm 4. In random minimum cost method (RaMCM), both the variable costs and the fixed costs are considered based on linear relaxation discussed in 3.2. Depending on the size k of the problem, a set of cells with minimum costs is considered for allocation. To control the size k, a formula $k = n/5$ (rounded to the next decimal place) is considered.

A cell within the k minimum cells are randomly selected and assigned a possible maximum shipment m_{ij} to the selected cell (i, j). The row (column) corresponding to cell (i, j) is eliminated if its supply (demand) is satisfied. If both row and column are simultaneously satisfied, then only one is eliminated. Recalculate the transformed cost matrix based on the linear relaxation after every assignment. Then identify k minimum cells and randomly choose the next cell to be allocated. Adjust the supply and demand for those rows and columns which are not crossed out.

This process is repeated until exactly one row (column) is left for allocation. When exactly one

Algorithm 3: Relaxed minimum cost algorithm

Input : Variable costs C, supplies $\underline{s}$ and demands $\underline{d}$.
Output: Initial basic feasible solution (bfs).

$\gamma,\ \lambda \leftarrow n \times m$ empty matrix;
for *all empty cells in* γ **do**
    // Determining the Balinski transportation matrix
    $\gamma_{ij} \leftarrow c_{ij} + f_{ij}/m_{ij}$;
end
while *all supplies* $\underline{s}$ *and demands* $\underline{d}$ *are not satisfied* **do**
    search for a cell with $s_i,\ d_j > 0$ *and the minimum cost*;
    $i \leftarrow$ row number of cell with $\min(\gamma_{ij})$;
    $j \leftarrow$ column number of cell with $\min(\gamma_{ij})$;
    $\Delta = s_i - d_j$; // difference of supply i and demand j
    if $\Delta \geq 0$ **then**
        // open route ij and assign possible maximum flow to it
        $\lambda_{ij} = d_j$;
        $s_i = s_i - d_j$; // adjust supply i and demand j
        $d_j = 0$;
    else
        $\lambda_{ij} = s_i$;
        $d_j = d_j - s_i$; // adjust supply i and demand j
        $s_i = 0$;
    end
end
return: λ

Algorithm 4: Random minimum cost algorithm

Input : Variable costs C, Supplies $\underline{s}$ and demands $\underline{d}$.
Output: Initial basic feasible solution (bfs).

$\gamma,\ \lambda \leftarrow n \times m$ empty matrix ;
for *all empty cells in* γ **do**
    // Determining the Balinski transportation matrix
    $\gamma_{ij} \leftarrow c_{ij} + f_{ij}/m_{ij}$;
end
while *all supplies* $\underline{s}$ *and demands* $\underline{d}$ *are not satisfied* **do**
    randomly select cell (i, j) *with* $s_i, d_j > 0$ *from the 20% lowest cost cells*;
    $\Delta = s_i - d_j$; // difference of supply i and demand j
    if $\Delta \geq 0$ **then**
        // open route ij and assign possible maximum flow to it
        $\lambda_{ij} = d_j$;
        $s_i = s_i - d_j$; // adjust supply i and demand j
        $d_j = 0$;
    else
        $\lambda_{ij} = s_i$;
        $d_j = d_j - s_i$; // adjust supply i and demand j
        $s_i = 0$;
    end
end
return: λ

row or column is left, all the remaining variables are basic and are assigned the only feasible allocation. When the supply and demand constraints are satisfied, an initial basic feasible solution is obtained. The solution to be feasible must consists of $n + m - 1$ feasible cells.

Maximum flow minimum cost method

In the Maximum Flow Minimimum Cost Method (MaxFlowMCM), the costs (variable and fixed costs) were weighted differently and compared based on an adjustment factor, α, which varies from 0.1 to 0.9. Experimentation have shown that the method perform well with the adjustment factor, $\alpha = 0.2$. Thus, the algorithm was run with $\alpha = 0.2$ and the results was considered for improvement.

As presented in Algorithm 5, it starts by determining the possible maximum allocation that can be made in every unoccupied cell. The transformation is then done based on the adjustment factor $\alpha = 0.2$, taking into consideration the possible maximum allocation in every unoccupied cell. After all the unit costs are determined in every unoccupied cell, the algorithm allocates the maximum possible flow to the cell with minimum transformed cost. All supplies and demands are adjusted based on the allocation and the process gets repeated until all supplies and demands are satisfied. Once all supplies and demands are satisfied, a basic feasible solution is obtained.

Algorithm 5: Maximum flow minimum cost selection criterion

Input : Variable costs C, Supplies $\underline{s}$ and demands $\underline{d}$.
Output: Initial basic feasible solution (bfs).

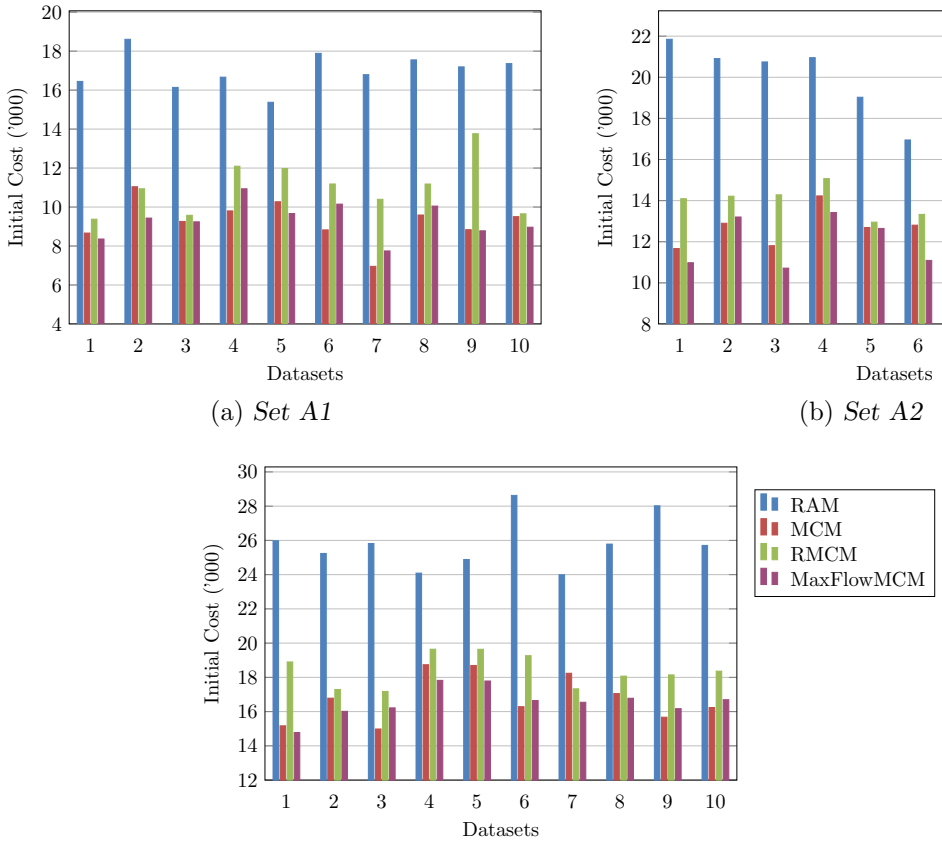
β, $\lambda \leftarrow n \times m$ empty matrix ;
while *all supplies* $\underline{s}$ *and demands* $\underline{d}$ *are not satisfied* **do**
    // Determining the maximum flow transportation matrix
    for *all empty cells in* β **do**
        $\beta_{ij} \leftarrow \alpha \times c_{ij} \times m_{ij} + (1 - \alpha) \times f_{ij}$;
        // where $m_{ij} = min(s_i, d_j)$
    end
    sort all elements in β *from small to highest*;
    $i \leftarrow$ row number of cell with value minimum β_{ij};
    $j \leftarrow$ column number of cell with value minimum β_{ij};
    $\Delta = s_i - d_j$; // difference of supply i and demand j
    (open route ij *and assign possible maximum flow to it)*;
    if $\Delta \geq 0$ **then**
        $\lambda_{ij} = d_j$;
        $s_i = s_i - d_j$; // adjust supply i and demand j
        $d_j = 0$;
    else
        $\lambda_{ij} = s_i$;
        $d_j = d_j - s_i$; // adjust supply i and demand j
        $s_i = 0$;
    end
end
return: λ

4.3.2 Computational results of initial solution methods

Most previously reported research on fixed charge transportation problems has also indicated that computational efficiency is highly effected by the relative size of the fixed and variable costs. Thus, it is essential to trade-off the fixed and variable costs into a single unit cost, by relaxing the integer restriction on binary variable y_{ij}, as discussed in Section 3.2. The Balinski [8] matrix is computed and the four heuristic methods of determining an initial solution applied. The results are graphically presented in Figure 4.2.

Figure 4.2 shows the initial solutions based on Set A1 (when $\theta = 0.0$), Set A2 (when $\theta = 0.2$) and Set A3 (when $\theta = 0.5$) of Dataset 1, which shows that the MCM, RMCM and the MaxFlowMCM

perform more or less the same. The RAM performs far worse than rest of the methods, thus it cannot be considered.



(a) *Set A1* (b) *Set A2*

(c) *Set A3*

FIGURE 4.2: *Initial solutions of different heuristic criteria on Dataset 1.*

The fixed charge transportation problem is known to be NP-Hard [19]. Thus, the difficulty of the problem increases with the size of the problem. This being the case, it is important to determine whether a specific instance of the problem is solvable in a reasonable amount of time. Since the three methods averagely performs the same, computational time was then taken into account. The method that generate an acceptable (less transportation costs) starting solutions with less computational time will be considered as a good approach to find initial solutions.

Table 4.2 shows average computational runtime taken to generate the initial solutions using the above mentioned heuristic approaches. On average, the MaxFlowMCM outperforms all the methods. The next step is to improve the obtained initial solution to see how close it can get to the optimum. Good initial solutions may provide better final solutions after improvements.

Dataset	RAM	MCM	RMCM	MaxFlowMCM
Set A1	0.0287	0.0286	0.0296	0.0268
Set A2	0.0290	0.0296	0.0312	0.0273
set A3	0.0324	0.0294	0.0291	0.0275
Average	0.0292	0.0300	0.0300	0.0272

TABLE 4.2: *Average computational (in seconds) time taken to generate the initial solutions using four heuristic methods on Dataset 1.*

4.4 Solution improvement

There is no simple test for optimality for FCTP. Searching for better solutions involves analysis of each unused (open) cell to determine the potential for reducing the total cost of the solution. This is accomplished by transferring one unit into an empty cell and noting its impact on costs. If costs are increased, that implies that using the cell would increase total costs. If costs remain the same, that implies the existence of an alternative option with the same total cost as the current plan. However, if analysis reveals a cost decrease, the implication is that an improved solution is possible. In conjunction with the tabu search algorithm, which will serve as a memory for cells that had already been visited, the stepping stone method will be used as a tool of solution improvement.

4.4.1 Stepping stone method

The stepping stone method is used to improve a current initial feasible solution obtained by any of the described heuristic methods. Thus, the stepping stone method is a procedure for finding the potential of any non-basic variables (open cells) in terms of the objective function. Through the stepping stone method, the effect on the transportation cost in case one unit is assigned to the open cell is determined.

An optimal solution may be obtained by making successive improvements to an initial basic feasible solution until no further decrease in the transportation cost is possible. Figure 4.3 presents the improvement process using the stepping stone method, which can be broken down as presented in Algorithm 6. The process starts with the initial feasible solution obtained using the MaxFlowMCM. This solution is revised until the stopping criteria is met, then it returns the best solution obtained.

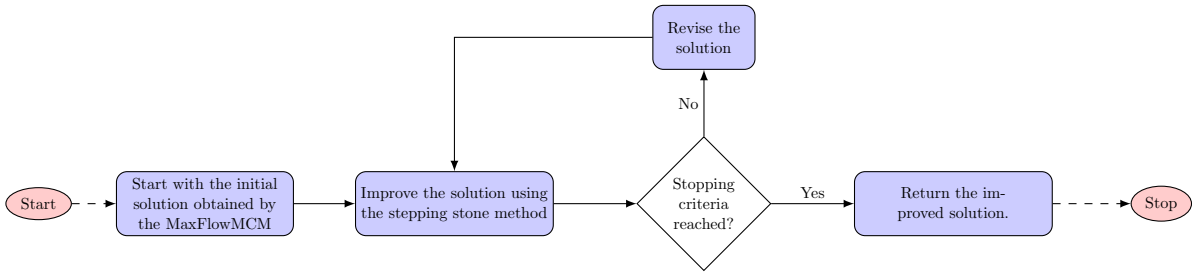


FIGURE 4.3: *The flow chart showing the improvement process using the stepping stone method*

The stepping stone method makes use of an unused or vacant cell as a point of destination to evaluate if the current solution can still be improved. The general process is to look for at least three occupied cells, rectangular in position to the point of destination. The movement is vertical and horizontal. The method derives its name from the analogy of crossing a pond using stepping stones. The occupied cells are analogous to the stepping stones, which are used in making certain movements in this method. An ordered sequence of at least four different cells is called a *loop* if any two consecutive cells lie in either the same row or same column.

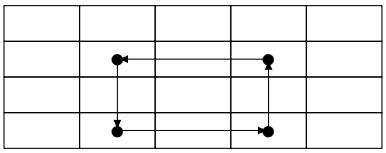
In the definition of a loop, the first cell is considered to follow the last cell, so the loop may be thought of as a closed path (as shown in Table 4.3). Table 4.3a and 4.3b shows the appropriate rectangular loops accepted when using the stepping stone method. Table 4.3c does not present a rectangular loop, so it is not accepted when improving the transportation problem. No three consecutive cells lie in the same row or column (i.e. Table 4.3d) and the last cell in the sequence has a row or column in common with the first cell in the sequence.

Algorithm 6: Stepping stone algorithm

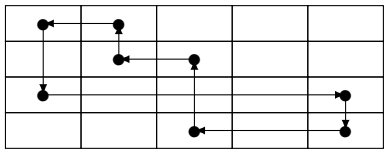
Input : initialSolution T, initialCost C_0, $\underline{s}$,$\underline{d}$, C and F
Output: bestSolution

```
currentSolution ← T;
bestSolution ← T;
bestCost ← C_0;
StopCriteria ← False;
while Stopping criteria is not True do
    for all empty cells in a matrix do
        (determine a close path, open route to determine newSolution);
        if newSolution improves the currentSolution then
            currentSolution ← newSolution;
            if ObjValCal(newSolution) < bestSolution then
                bestSolution ← newSolution;
                bestCost ← ObjValCal(newSolution,s,d,C,F);
            end
        end
        if currentSolution = bestCost then
            open the least worsening route;
            StopCriteria ← False;
        else
            StopCriteria ← True;
        end
    end
end
return: bestSolution
```

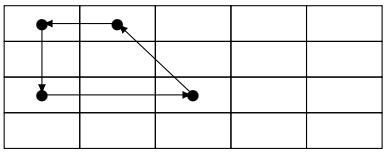
TABLE 4.3: *Figures showing acceptable and unacceptable stepping stone loops*



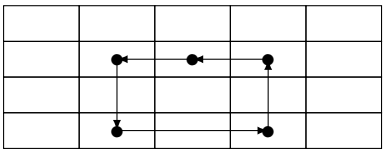
(a) *Loop 1 (feasible)*



(b) *Loop 2 (feasible)*



(c) *Loop 3 (infeasible)*



(d) *Loop 4 (infeasible)*

The necessary condition for the optimality is that the number of occupied cells is exactly equal to $m + n - 1$, where m is the number of rows and n is equal to the number of columns [41]. The process of identifying feasible paths for any given initial solution involves all the evaluation of the empty (unoccupied) cells within the solution matrix. Starting with any unused cell, a closed path is traced back to its origin through cells that are currently allocated – considering vertical and horizontal moves. Depending on the variation of the variable costs and the fixed costs, it may not be necessary to evaluate every open cell in a matrix. Not every cell brings an improvement to the objective function value. Evaluating every open cell may prolong the algorithm's computational time, thus – one may think of different selection strategies of cells to be evaluated.

4.4.2 Selection criteria

At each iteration of the stepping stone approach, the algorithm evaluates open cells to identify which ones makes an improvement to the objective function value when they are utilised. Four selection criteria were considered to choose cells to be evaluated.

Row-column random selection criterion

In this approach, on each iteration the objective function value is saved for further computations. As shown in Figure 4.4, the algorithm sequentially looks for a cell with minimum cost in each row, evaluate and accepts its solution if it makes an improvement to the objective function value. Else the algorithm iterate in each column sequentially and evaluate cells with minimum costs and accept its solution if it makes an improvement to the current solution. However, if an improvement cell is not found in both the rows and columns evaluation, the algorithm randomly choose any open cell and accept its solution. This process is repeated until the stopping criteria is met.

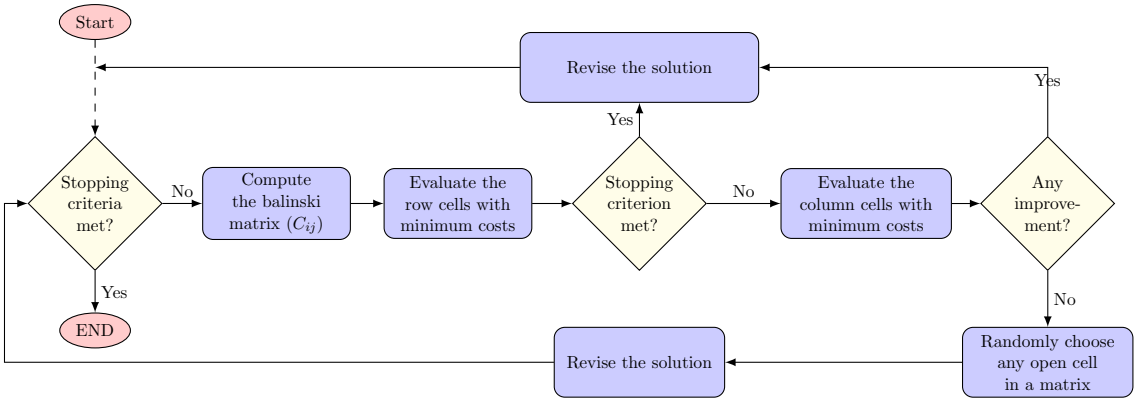


FIGURE 4.4: *Flow chart showing the row-column random selection criterion*

Row-column selection criterion

This procedure does not differ much from the row-column random selection criterion, except that randomness is not considered. In this approach a cell with minimum relaxed cost in each row is selected, until a cell that makes an improvement to the current solution is reached. Once a cell with an improved solution is reached the algorithm considers the solution and start the evaluation process again. If all the rows to the problem are visited without any improvement to the current solution, the algorithm then start the column evaluation by evaluating a cell with a minimum relaxed cost. Should a cell be found with an improved solution in the column evaluation, its solution is considered and the algorithm restarts the row evaluation process. If each row and each column has been evaluated without any improvement, amongst all the evaluated cells, a solution with the least worsening move (minimum objective function value) is considered and the process is restarted (as shown in Figure 4.5). This process is repeated until the stopping criteria is met, which is the number of iterations completed without further improvements to the objective function value.

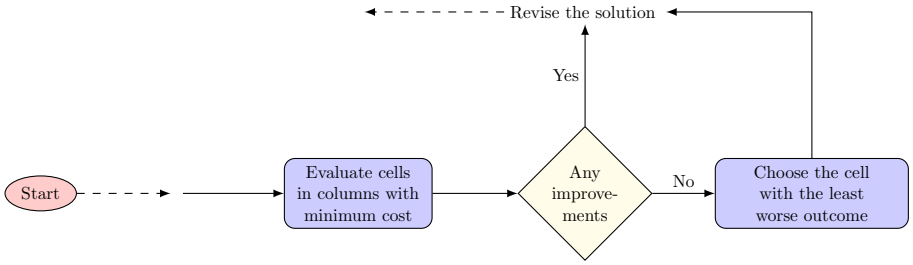


FIGURE 4.5: *Flow chart showing the row-column selection criterion*

Sequential criterion

In this approach, the algorithm iterate sequentially on all open cells until a cell that makes an improvement to the objective function value is reached. Once the cell is reached, its solution is accepted (as shown in Figure 4.6). The algorithm then recalculate the new transformed costs and restart the process of cell evaluation. At each iteration, objective function values are compared and the solution with a least objective function value is saved. If all cells are evaluated and none of the cells makes an improvement to the objective function value, the algorithm considers the cell that makes the least worsening move and restart the process by computing the new matrix.

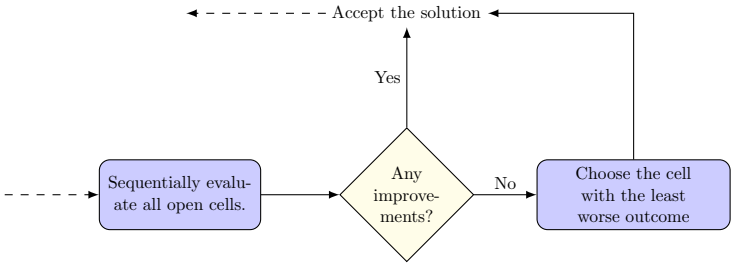


FIGURE 4.6: *Flow chart showing the sequential selection criterion*

Random selection criterion

This approach is more or less similar to the sequential criterion, except that there is randomness in it. This algorithm also evaluates all open cells sequentially until a cell that makes an improvement to the objective function is reached. If there is a cell that improves the current solution, its solution is accepted and the algorithm recalculate the relaxed cost matrix and start the evaluation. Should all open cells be evaluated without any improvement to the objective function value, a random choice to the cells is considered (as shown in Figure 4.7), irrespective of how bad its solution is.

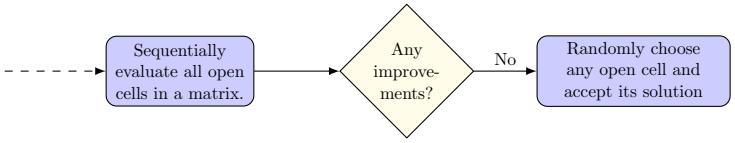


FIGURE 4.7: *Flow chart showing the random selection criterion*

All the selection criteria have been tested on the instances described in Section 3.1. Only one selection criterion can be considered when improving the current initial solution. If all cells have been evaluated for improvement and no improvement is possible, the problem is either stuck in

local optimum or an optimal solution has been reached. While accepting worse moves, a list is considered to store the indices of the cell that have just been closed. This list is called the tabu list, which serves as a memory for cells that should not be visited again. The cells remain in the tabu list until another move (improvement or worse) is accepted.

4.5 Tabu list

A tabu list is a way of maintaining a list of solution points that must be avoided (not allowed) or a list of move attributes that are not allowed. At each iteration of TS, the short-term memory is updated. It is necessary to check at each iteration if a generated solution does not belong to the list of recently visited solutions. The tabu list usually contains a constant number of tabu moves. Usually, the attributes of the moves are stored in the tabu list. The main goal of using the short-term memory is to prevent the search from revisiting previously visited solutions which leads to cycles. As mentioned, storing the list of all visited solutions is not practical for efficiency issues. The length of the tabu list is also important; the longer the tabu list length, the more likely it is to escape from these local minima, but also the less thoroughly will each one be searched [14]. If the tabu list is too long, the oldest candidate solution is removed and it is no longer taboo to reconsider [26].

The tabu list is used to store the routes that are closed in each iteration until another move is made. The list are cleared, but the last route that was stored is kept to avoid an immediate circle. The length of the tabu list, l is determined by the size of the problem, $n = m$. For small problems $l = n/2$ was used when $n < 30$, while $l = n/3$ for larger problems with $n \geq 30$. It has been shown, however, that fixed-length tabu lists cannot always prevent cycling, and some authors have proposed varying the tabu list length during the search [14]. The main approach to TS is to maintain a tabu list L, of some maximum length l, of candidate solutions. Whenever a new candidate solution is adopted, it goes in the tabu list. Because it is difficult to determine the optimality of a solution to a FCTP, it is wise to have another stopping criteria in an algorithm.

4.6 Aspiration criteria

Aspiration criterion is used to make the search process finite. It allows for exceptions from the tabu list, if such moves lead to promising solutions. In theory, the search could go on forever, unless the optimal value of the problem at hand is known beforehand. In practice, the search has to be stopped at some point. The termination criteria used in this algorithm are:

- after a number of iterations (or a fixed amount of CPU time);
- after a number of iterations without an improvement in the objective function value;
- when the objective reaches a pre-specified threshold value.

In complex tabu schemes, the search is usually stopped after completing a sequence of phases, the duration of each phase being determined by one of the above criteria. If all cells have been tested for improvement and the optimal solution has not been reached, the problem has stuck in a local optimum. A local optimum is the best solution to a problem within some neighbourhood of possible solutions. This concept is in contrast to the global optimum, which is the optimal solution when every possible solution is considered. The methods applied above uses the primal approach to the FCTP.

The algorithms were also executed on LINGO solver. This software uses the exact optimisation approach which guarantees optimal solutions. The algorithms were only ran on Dataset 1, because of the dimensions of the problem instances. The algorithms were implemented and coded in LINGO [34] and has been run on the same computer. LINGO solver is very useful to solve hard optimisation problems. It solves the problems by using branch and bound methodology and can be used to verify and compare the results with the traditional and meta-heuristic optimisation methods. Although the CPU time to find optimal solutions using LINGO is very much slower when compared to the time taken by heuristic procedures, particularly NP-hard problems, it performs better in terms of solution quality. Furthermore, the problem was solved by means of a primal-dual approach.

4.7 A primal-dual method for solving transportation problems

The primal-dual method (PDM) is described using arrays for ease of understanding, but computer implementations are usually based on the network formulation. This saves in memory requirements and running time for solving practical problems which are almost always sparse. The PDM is related to the Dantzig, Ford, and Fulkerson primal-dual algorithm [13]. It is analogous to their method because both methods minimize the sum of the artificial variables in the primal problem while preserving the conditions of complementary slackness [45]. This method proceeds by maintaining a feasible solution to the dual problem without necessarily satisfying the primal restrictions. The values of the dual variables are adjusted, preserving dual feasibility but moving toward primal feasibility. Once primal feasibility has been achieved, the optimal solution is attained. This is applicable only on standard transportation problems. In this thesis, once primal feasibility has been achieved, an initial basic solution is attained. The dual problem may be formulated as

$$\begin{aligned} \text{maximize } & \sum s_i u_i + \sum d_j v_j \\ \text{subject to } & u_i + v_j \leq c_{ij} \qquad i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \\ & u_i, v_j \geq 0 \qquad i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \end{aligned} \tag{4.13}$$

where s_i is the supply at origin i and d_j is the demand at destination j. Let $\bar{c}_{ij} = c_{ij} - u_i - v_j$, for $i = 1$ to n, $j = 1$ to m. These are the **reduced cost** with respect to (u, v), and (u, v) is dual feasible if and only if they are all non-negative. The complementary slackness conditions for optimality in these problems are

$$x_{ij}\bar{c}_{ij} = 0 \qquad i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \tag{4.14}$$

where x_{ij} represent the number of units shipped from source i to destination j. The cell (i, j) in the array is said to be a feasible cell with respect to (u, v) if $\bar{c}_{ij} = 0$; otherwise it is infeasible. The dual variable u_i is the marginal cost of making a shipment of one unit from the origin i, and v_j is the marginal cost of making a shipment of one unit into the destination j, while $\bar{c}_{ij}$ is the marginal cost (opportunity cost) of making a shipment of one unit from the origin i into the destination j. That means only the arcs from the origin i to the destination j with zero opportunity (marginal) cost are allowed to have positive amount of shipments. Hence these arcs are called the feasible arcs.

The network obtained by deleting all the infeasible arcs, is known as the feasible or equality subnetwork with respect to the marginal costs (u, v). The complementary slackness conditions

require that the flow amounts should be 0 on all the infeasible arcs. The flow problem, known as the restricted primal at this stage, is to find a maximum value flow from the super source to the super sink in the equality subnetwork. Let S be the set of all feasible cells (i, j). It is equivalent to

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\epsilon S} (x_{ij} \ : \ \text{over } (i,j) \text{ feasible}) \\ \text{subject to} \quad & \sum_{j=1}^{m} x_{ij} \ \leq \ s_i, \ i = 1 \text{ to } n \\ & \sum_{i=1}^{n} x_{ij} \ \leq \ d_j, \ j = 1 \text{ to } m \\ & x_{ij} \ \geq \ 0, \text{ if } (i,j) \ \in S, \ 0 \text{ otherwise} \end{aligned} \tag{4.15}$$

The primal-dual algorithm maintains vectors $\underline{x}$, (u, v) which always satisfy the constraints in (4.13), (4.14), (4.15). When the vector $\underline{x}$ satisfies the equality constraints in (3.4), it is an optimum solution and the method terminates. An important benefit of this method is that degeneracy causes no problems.

An algorithm for finding a set of feasible cells satisfying the necessary conditions

The algorithm that follow does not consider the primal solution. It acts only on the basis of the information obtained from the dual variables and conditions (4.15) when they do not hold. It consists of changing iteratively the dual variables (without regard for the primal), while maintaining their feasibility, until they satisfy the required conditions. Given a bipartite network whose node set $I \cup J$ is partitioned into n origin nodes ($I = \{i_1, \ldots, i_n\}$) and m destination nodes ($J = \{j_1, \ldots, j_m\}$), an algorithm that finds a set of feasible cells is as follows:

Step 0 **Initialization phase**:
$u_i \leftarrow \min_{i \in J} c_{ij}$ for all $i \in I$;
$v_j \leftarrow \min_{j \in I}(c_{ij} - u_i)$ for all $j \in J$;
List $= \emptyset$. Define $x^1 = 0$.

Step 1 **Tree growth routine**: Let $\underline{x} = x_{ij} \ \forall_{ij}$ be the present flow

Substep 1.1 : Label each row i satisfying $\sum_j \bar{x}_{ij} < s_i$ with $(s, +)$, and include in the list.

Substep 1.2 : If list $= \emptyset$, tree growth has terminated and there is a non-breakthrough. The present flow is maximum value flow in the feasible sub-network, go to Step 3. Otherwise, select a row (column) from the list for scanning and delete it from the list.

Forward labelling Scanning row i consists of labelling each unbalanced column j such that (i, j) is a feasible cell, with the label (row i, $+$).

Reverse labelling To scan column j, label all unbalanced row i satisfying $\bar{x}_{ij} > 0$ with (column $j, -$).

If any column without all allocation has been labelled, there is a breakthrough, go to Step 2. Otherwise, include all newly labelled rows and columns in the list, repeat Substep 1.2.

Step 2 **Flow change routine**: Suppose column j satisfying $\alpha = b_j - \sum_i \bar{x}_{ij} > 0$ has been labelled. Trace its predecessor path using the labels that ends at row i with label of $(s, +)$, and allocate the signs as in stepping stone algorithm. Determine the minimum entry having a minus sign, add it to the plus values and subtract it from the minus values. Increase flow from column j through the path to row i as much as possible without violating feasibility, erase the labels on all the rows and columns and go to Step 1

Step 3 **Dual solution change routine**: Compute σ, the minimum value of reduced cost coefficient among cells in labelled rows and unlabelled columns. This σ will be greater than zero ($\sigma > 0$). If $\sigma = +\infty$; this can only happen if some x_{ij} are constrained to be 0 in the problem. Thus, there is no feasible solution and the algorithm terminates. If σ is finite, add it to the value of u_i in all labelled rows and subtract it from the value of v_j in all labelled columns. Compute the new reduced cost coefficient in each cell. Retain the present labels on all the labelled rows and columns, but include all the labelled rows in the list, and resume tree growth by going to Substep 1.2 in Step 1.

After the algorithm finishes, the flows from source to destination need to be feasible. For the primal-dual algorithm degeneracy is not an concern since the algorithm does not seek basic feasible solutions, but feasible cells.

Solution improvement

Once a basic feasible solution is attained an appropriate methods of solution improvement are applied, as discussed in Section 4.4. Two methods of solution improvement are considered in this thesis. The methods are 1) the stepping stone method and 2) the break-and-fix method. Both the two methods uses the tabu search algorithm.

The stepping stone method

In this method, each open cell is evaluated for solution improvement and the cell gets shipment allocation if it makes improvement to the current solution. This method is applied exactly as in Section 4.4.1 and it uses the *sequential selection criterion*, where if all cells are evaluated with none making an improvement to the solution, a least worsening move is made to the current solution.

Break-and-fix method

In this method, the primal feasibility is violated by identifying the most expensive feasible arc (i, j) and make it infeasible. This results with an allocation of zero shipment to arc i, j and the shipment amount reversed to the corresponding supply i and demand j. The algorithm then assign a possible maximum cost to the Balinski matrix cell (i, j) to avoid assignment to the same cell that just left the solution. Once the feasibility is broken and supply i together with demand j are not satisfied, the algorithms then goes back to Step 1 of the the primal-dual algorithm discussed in Section 4.7.

The method of breaking feasibility is repeated for a number of iterations until a best solution is attained. The solution cost will then be compared to the optimal solution cots, *if exists*, and the solution gap be determined to see how close the obtained solution is to the optimal. The primal-dual method guarantees an optimal solution only to the standard transportation problem and not to fixed charge transportation problems.

4.8 Chapter summary

Mathematical models and the description of all parameters has been presented in this chapter. Different heuristics for determining the initial solutions was presented as well as their computational performances. Thus, a good initial solution was chosen based on its performance (i.e. solution quality and computational time). Lastly, the master algorithm (tabu search) and the stepping stone method are discussed. This will assist in improving the current solution.

CHAPTER 5

Computational experiments

Contents

5.1 Stepping stone method with a greedy local search 42
5.2 Computational results on different selection criteria 43
5.3 Stepping stone method with tabu search algorithm 45
5.4 Lingo performance analysis 47
5.5 Experimental analysis on Dataset 2 48
5.6 The primal-dual algorithm solution experiments 50
5.6.1 Initial feasible solution *50*
5.6.2 Solution improvement *50*
5.7 Chapter summary 52

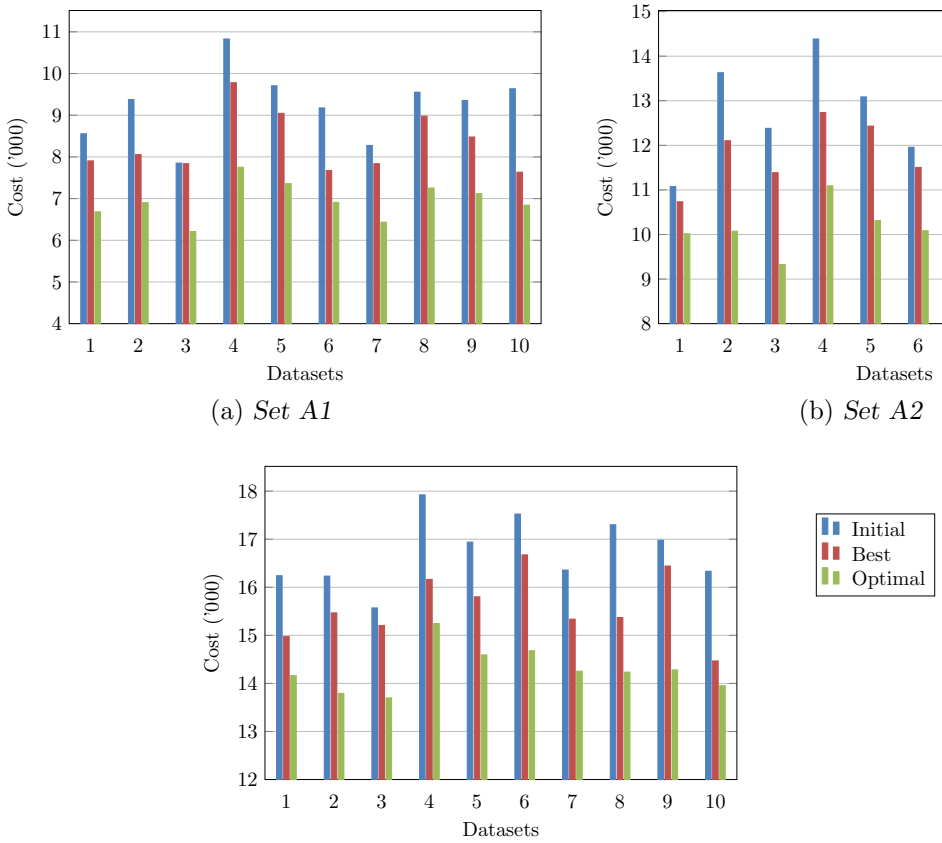
Different heuristic methods for solving the fixed charge transportation problems were discussed briefly in Chapter 4. In this chapter methods are numerically evaluated and analysed based on their performance. A series of experiments are reported which determines the good initial solutions and the improved solutions that may be optimal or near optimal solutions. A primal approach was used, with the stepping stone method, to improve our initial solutions in conjunction with a tabu search algorithm. Also, a dual approach was implemented, called the primal-dual method, for the fixed charge transportation problem. A tabu search algorithm was also considered in this approach. All instances used in this thesis fulfil the assumption made in Section 1.1 that, $\sum_{i=1}^{n} s_i = \sum_{j=1}^{m} d_j$ (i.e. instances are already *balanced*). It is necessary to prepare initial feasible solutions, which may be done in several different ways; the only requirement is that the destinations needs be met within the constraints of source supply.

Initial feasible solutions involve assigning flows to cells to satisfy supply and demand constraints. Each basic feasible solution of the FCTP includes $m+n-1$ basic variables, and each variable represents the amount shipped from one origin to one destination. Different heuristics were considered and compared to determine the better initial solutions. A better method (MaxFlowMCM) is observed and its starting solutions will be considered for improvement. An optimal solution is one where there is no other set of transportation routes that will further reduce the total transportation cost. Thus each open cell is evaluated in the transportation matrix in terms of an opportunity of reducing total transportation cost. The chapter starts by discussing and presenting the results to the stepping stone approach with tabu list length, $L = 0$. Then later the tabu list length will be adjusted, based on the problem size, to check if there exist further improvements to the problems. When the tabu list length is zero ($L = 0$), the solutions may be considered as greedy local search.

5.1 Stepping stone method with a greedy local search

Once a good initial solution is calculated, the solution must be improved until a stopping criterion is reached. The initial solution used as a starting point in this chapter is from the MaxFlowMCM since it had the minimum total transportation costs within a minimum computational time over all tested heuristics. The stepping stone method was used to determine if there are better solutions as shown in Section 4.3.2.

Figure 5.1 shows the solution improvement when using the stepping stone method on Dataset 1 of the problem instances. Although optimal solution have not been reached, there exists an improvement on all instances of Set A1, Set A2 and Set A3. There was a slight improvement on problem instance 3 on Set A1 which cannot clearly be seen from Figure 5.1(a), but can be seen in Table A.4. The stepping stone method has a trend of getting stuck in suboptimal regions or allows circulation of solutions without any improvement to the objective function. To avoid cycling, a tabu search algorithm may be used to improve a current solution.



(a) *Set A1*

(b) *Set A2*

(c) *Set A3*

FIGURE 5.1: *Solution improvement of the MaxFlowMCM using the stepping stone method when the greedy local search is considered on Dataset 1.*

The implementation of tabu search uses memory structure that describe the visited solution or user-provided sets of rules. If a potential solution has been previously visited within a certain short-term period or if it has violated a rule, it is marked as *tabu* (forbidden) so that the algorithm does not consider that possibility repeatedly. In Chapter 4 various types of selection criteria are discussed. These are methods of selecting cells to be evaluated for solution improvement. When considering the selection criteria, the tabu list length were adjusted based on the size of

the problem. Instances in Dataset 1 were all evaluated considering various selection criteria.

5.2 Computational results on different selection criteria

Table 5.1 summarises the best results obtained using different selection criteria for small sized problem instances (Dataset 1). This results are based on Set A1. Column 1 of the table presents the problem number followed by its known optimal solution in column 2. The results to the first selection criterion (Row-column random selection) follows in columns 3, 4 and 5. The best obtained solution cost in column 4 and the CPU runtime taken to obtain the solution in column 4. It took an average of 0.06 seconds to find a solution, which resulted on an average solution gap of 23.87%. This results were outperformed by the row-column selection criterion presented in columns 6, 7 and 8. In this criterion, the algorithm obtained its best solution with an average of 22.93% solution gap to the optimal solution within 0.69 seconds of computational runtime. Although the algorithm obtained better results than the row-column random selection criterion, it was found to be computationally expensive, mainly because there was no randomness on this algorithm.

Next consider the random selection criterion on column 9, 10 and 11. This algorithm outperformed the row-column random selection and the row-column selection criteria in terms of solution quality. This algorithm has an average of 20.94% solution gap to the optimal solution computed within 0.78 seconds of computational runtime. Lastly, the sequential selection criterion outperformed all the tested selection criteria based on Set A1 of the instances. Although the algorithm took more time to get to its best solution, an average of 11.47% solution gap to the optimal solution was obtained. It took an average of 2.30 seconds to get to this level of solution quality.

		Row-Column random			Row-column selection			Random selection			Sequential selection		
	Optimal	Best	Time	**SolGap (%)**	Best	Time	**SolGap (%)**	Best	Time	**SolGap (%)**	Best	Time	**SolGap (%)**
1	6683	7768	0.049	**16.24**	7366	1.004	**10.22**	8058	0.746	**20.57**	7136	2.648	**6.78**
2	6903	8056	0.085	**16.70**	8056	0.676	**16.70**	8056	0.780	**16.70**	7798	2.096	**12.97**
3	6210	7655	0.059	**23.27**	7565	0.888	**21.82**	7837	0.723	**26.20**	6608	5.199	**6.41**
4	7753	10266	0.095	**32.41**	10223	0.669	**31.86**	9780	0.859	**26.14**	8752	0.806	**12.89**
5	7360	9481	0.034	**28.82**	9481	0.597	**28.82**	9045	0.770	**22.89**	8332	0.896	**13.21**
6	6911	8875	0.045	**28.42**	8787	0.631	**27.15**	8746	0.745	**26.55**	7342	3.640	**6.24**
7	6434	7436	0.069	**15.57**	7436	0.636	**15.57**	7837	0.769	**21.81**	7727	0.420	**20.10**
8	7254	9443	0.049	**30.18**	9236	0.624	**27.32**	8045	0.910	**10.90**	8138	2.489	**12.19**
9	7119	8278	0.100	**16.28**	8478	0.627	**19.09**	8478	0.736	**19.09**	8153	4.611	**14.52**
10	6843	8950	0.042	**30.79**	8950	0.592	**30.79**	8113	0.769	**18.56**	7488	0.199	**9.43**
Average			0.06	**23.87**		0.69	**22.93**		0.78	**20.94**		2.30	**11.47**

TABLE 5.1: *Summary of best solution obtained using different selection criteria on Set A1*

The same selection criteria was tested on Set A2 of problem instances and the results are summarised in Table 5.2. The second column presents the optimal solution to the problems defined in Column 1. Using the row-column random selection criterion, the algorithm obtained an average of 16.55% solution gap to the optimal solution. This solution was obtained within an average of 0.07 seconds. Although the row-column selection criterion has no randomness, its solution does not differ much from the row-column random selection criterion. The row-column selection criterion has an average solution gap of 16.13% which was obtained with 0.41 seconds of computational time.

It is not clear which criterion performs the best since they evaluation is based on both the computational time and the solution quality. Assessing the random selection criterion, the

solution quality also does not differ much from the row-column random selection and row-column selection criteria. This algorithm obtained its best solution with an average of 16.59% solution gap to the optimal solution within an average of 0.35 seconds of computational runtime. Again, it is difficult to rate its performance in comparison to the other tested selection criteria. Lastly, the sequential selection criterion outperformed all the selection criteria in terms of solution quality. Computationally the algorithm is expensive, but it managed to obtain a solution with an average of 8.00% solution gap to the optimal solution within an average of 2.14 seconds.

		Row-Column random			Row-column selection			Random selection			Sequential selection		
	Optimal	Best	Time	**SolGap (%)**	Best	Time	**SolGap (%)**	Best	Time	**SolGap (%)**	Best	Time	**SolGap (%)**
1	10017	11029	0.039	**10.10**	11029	0.325	**10.10**	10734	0.286	**7.16**	10231	0.210	**2.14**
2	10075	13043	0.067	**29.46**	12734	0.353	**26.39**	12458	0.361	**23.65**	11157	0.763	**10.74**
3	9327	11051	0.080	**18.48**	11051	0.397	**18.48**	11387	0.300	**22.09**	9981	4.242	**7.01**
4	11093	13310	0.088	**19.99**	13186	0.366	**18.87**	13233	0.300	**19.29**	11408	0.739	**2.84**
5	10312	12592	0.060	**22.11**	12592	0.326	**22.11**	11883	0.387	**15.23**	11567	1.090	**12.17**
6	10086	11957	0.060	**18.55**	11437	0.606	**13.39**	11375	0.448	**12.78**	11170	4.678	**10.75**
7	9913	10474	0.082	**5.66**	11437	0.358	**15.37**	11716	0.290	**18.19**	10393	3.959	**4.84**
8	10495	11812	0.079	**12.55**	11795	0.525	**12.39**	12347	0.309	**17.65**	11566	2.982	**10.20**
9	10137	11456	0.039	**13.01**	11509	0.366	**13.53**	11258	0.456	**11.06**	11258	2.492	**11.06**
10	9939	11489	0.128	**15.60**	11002	0.501	**10.70**	11809	0.320	**18.81**	10759	0.245	**8.25**
Average			0.07	**16.55**		0.41	**16.13**		0.35	**16.59**		2.14	**8.00**

TABLE 5.2: *Summary of best solution obtained using different selection criteria on Set A2*

A summary of Set A3 problem instances based on different selection criteria is presented in Table 5.3. The optimal solution to this problem set is known and is presented in Column 2 with the corresponding problem numbers in Column 1. In this set, the sequential selection criterion outperformed all other selection criteria based on solution quality. The sequential selection criteria obtained near optimal solution to most tested problems and it manage to get optimal solution on problem instance 7 of Set A3. On average there is a 4.46% solution gap to the optimal solution, which was obtained within an average of 1.97 seconds of computational runtime.

That is followed by the random selection criterion with an average of 20.26% solution gap obtained within 0.77 seconds of computational runtime, then the row-column selection criterion with an average of 16.13% solution gap obtained within an average of 0.41 seconds of computational runtime. The row-column random selection criterion came in last with an average of 16.55% solution gap to the optimal solution obtained within an average of 0.07 seconds of computational runtime.

		Row-Column random			Row-column selection			Random selection			Sequential selection		
	Optimal	Best	Time	**SolGap (%)**	Best	Time	**SolGap (%)**	Best	Time	**SolGap (%)**	Best	Time	**SolGap (%)**
1	14161	15362	0.066	**8.48**	14974	0.680	**5.74**	14974	0.759	**5.74**	14351	3.123	**1.34**
2	13793	15178	0.065	**10.04**	15144	0.716	**9.79**	15467	0.747	**12.14**	15144	0.296	**9.79**
3	13699	15152	0.086	**10.61**	14889	1.230	**8.69**	15206	0.757	**11.00**	14234	1.675	**3.91**
4	15246	17044	0.072	**11.79**	17340	0.627	**13.73**	16573	0.813	**8.70**	16164	0.199	**6.02**
5	14593	16437	0.045	**12.64**	16186	0.612	**10.92**	15801	0.769	**8.28**	15432	0.257	**5.75**
6	14680	16597	0.042	**13.06**	16372	1.720	**11.53**	16674	0.775	**13.58**	15361	4.428	**4.64**
7	14255	15627	0.054	**9.62**	15724	0.597	**10.31**	15515	0.767	**8.84**	14255	2.688	**0.00**
8	14235	16664	0.085	**17.06**	16422	1.293	**15.36**	16370	0.780	**15.00**	14926	2.130	**4.85**
9	14281	16517	0.057	**15.66**	16518	0.687	**15.66**	16517	0.774	**15.66**	14935	4.822	**4.58**
10	13953	15909	0.057	**14.02**	15731	0.749	**12.74**	14468	0.801	**3.69**	14468	0.107	**3.69**
Average			0.06	**12.30**		0.89	**11.45**		0.77	**10.26**		1.97	**4.46**

TABLE 5.3: *Summary of best solution obtained using different selection criteria on Set A3*

From the above selection criteria analysis it can be observed that the *sequential selection criterion*

outperforms all other criteria. This selection criterion help the algorithm to escape local optima. Furthermore, the analysis comparing the performance of the algorithm is presented when the tabu list length $L = 0$ and $L > 0$. One major important aspect in tabu search algorithm is the tabu list length. In the computation, a variable tabu list length approach was used. With the exception of all problem instances, the tabu list lengths were allowed to vary in the following ranges: $10 \leq L \leq 15$ for small instances (Dataset 1) and $15 \leq L \leq 20$ for large instances (Dataset 2). In the tabu list cells are stored that had already been visited and did not make any improvement to the current solution. The tabu list is cleared only if there was a cell that made an improvement to the solution or the least worsening move has been accepted.

5.3 Stepping stone method with tabu search algorithm

The objective for the tabu search algorithm is to constrain an integrated heuristic from returning to recently visited areas of the search space, referred to as cycling. The strategy of the approach is to maintain a short term memory of the specific changes of recent moves within the search space and preventing future moves from undoing those changes [10].

Figure 5.2 shows the best solutions obtained when using the proposed algorithm when the tabu list length $L = 0$ and $L > 0$. This experiment was made to compare the performance of the algorithm when the tabu search algorithm is considered against the greedy local search algorithm. Figure 5.2(a) presents the graphical results of the algorithm based on Set A1. There is an improvement in all problem instances, although to some instances (i.e. Problem 2, 6, 7 and 10) there was a slight improvement. The results for Set A2, shown in Figure 5.2(b), shows that there is an improvement in all problems instances with a weak improvement on problem instance 9.

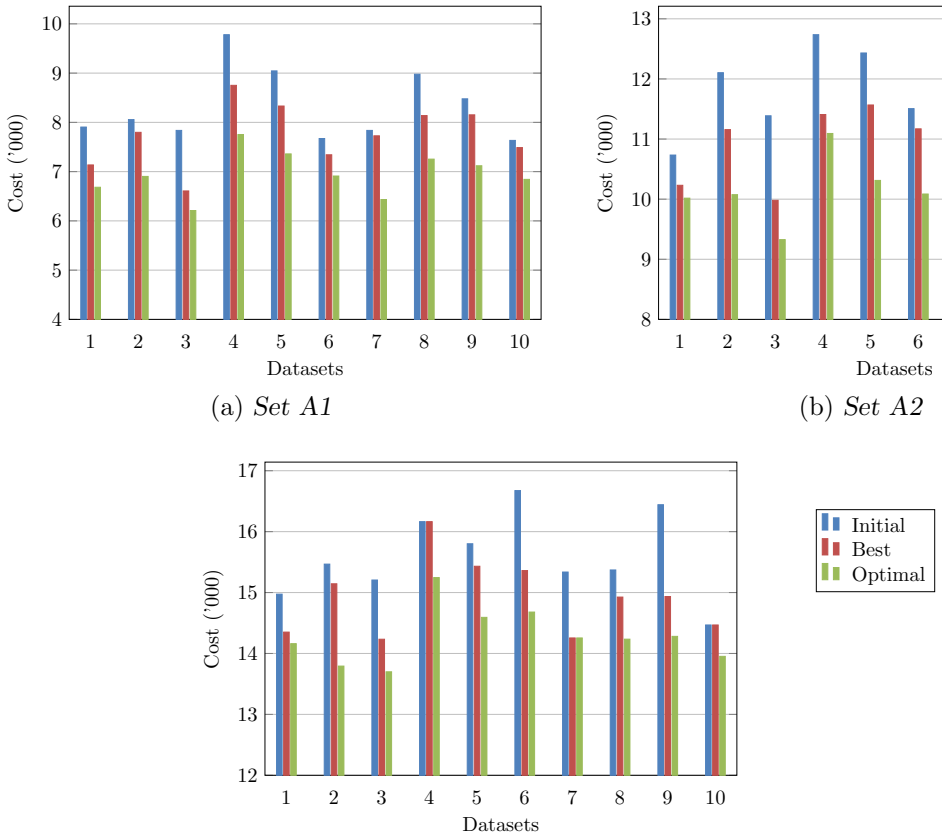
In Figure 5.2(c), with instances from Set A3, there are two problems (instances 4 and 10), which did not improve with a tabu list length $L > 0$. Although that being the case, the algorithm managed to obtain optimal solution to problem 7 of Class set 3 as shown in Figure 5.2(c). Despite the fact that the algorithm did not obtain optimal solution to all problem instances, it can be considered effective based on how close it was to the optimal level.

When analysing the solution gap to optimal solutions for both stepping stone method with tabu list length, $L = 0$ and stepping stone method with tabu list length, $L > 0$, it can be seen from Table 5.4 that the solution gets near optimal when tabu list length is longer than 0 ($L > 0$). Although there was not an improvement in all test instances after considering the tabu search algorithm to our algorithm, the solution gap to the optimal solution decreased from 14.65% to 9.98%. Based on this improvement, our algorithm is said to be effective.

Dataset	Greedy search	Tabu search
Set A1	19.75	11.47
Set A2	15.09	8.00
Set A3	9.12	4.46
Average	14.65	7.98

TABLE 5.4: *Average solution gap to the optimal solution for Dataset 1 problem instances.*

This computation experiment was designed to evaluate not only the performance of the proposed heuristic procedure for the fixed charge transportation problems on solution quality, but also on computational runtime. The computational runtime on both the stepping stone when the tabu



(a) *Set A1*

(b) *Set A2*

(c) *Set A3*

FIGURE 5.2: *Solution improvement of the MaxFlowMCM using the stepping stone method when tabu search algorithm is considered on Dataset 1 problem instances.*

list length $L = 0$ and $L > 0$ were measured.

The algorithm was run for 500 iterations and its rime was measured. This is reported in Table 5.5. Table 5.5(a) presents the computational runtime when 500 iterations were considered. After being able to find the best solutions obtained, this solution was used as a stopping criteria to measure the time the algorithm took to find the best solution. These computational times are reported in Table 5.5(b).

Dataset	Greedy search	Tabu search
Set A1	11.841	12.361
Set A2	12.066	12.287
Set A3	12.002	12.398
Average	11.969	12.349

(A) *Average CPU runtime using iterations as a stopping criteria*

Dataset	Greedy search	Tabu search
Set A1	0.097	2.35
Set A2	0.119	2.20
Set A3	0.124	2.03
Average	0.1136	2.26

(B) *Average CPU runtime using the known best solution as a stopping criteria*

TABLE 5.5: *Average computational time for the solution improvement of the MaxFlowMCM using the stepping stone method considering the greedy local search and the tabu search algorithm on Dataset 1.*

This experiment was done to observe if better results will be obtained if the algorithm is allowed to run for a longer period. Since the proposed algorithm does not have any randomness, it has been found that performing a large number of iterations has no significant effect in the solution

quality.

The solution movement was traced from the initial transportation cost to the best transportation cost. Should all the open cells be evaluated without any improvement, either an optimal solution is reached or the problem has stuck in a local optimum. Thus, a least worsening move is made in an attempt to escape the local optimum. Figure 5.3 shows the solution movement from the initial transportation cost to the optimal transportation cost where an optimal solution was attained. The analysis was made on Problem 7 of Set A3.

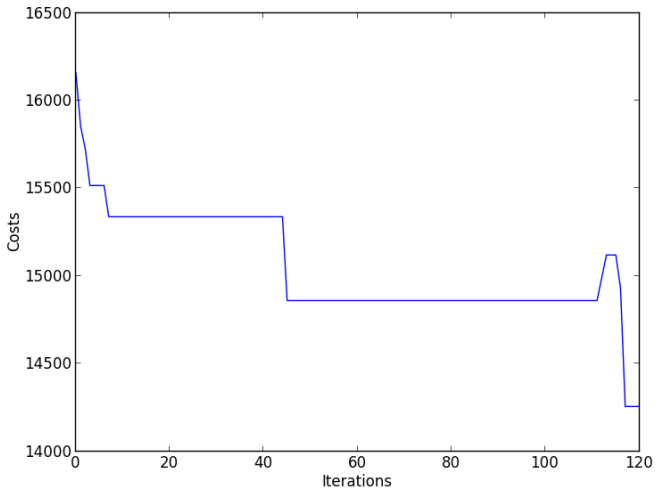


FIGURE 5.3: *Graphical representation of the solution movement when a least worsening move is made to escape the local optimum.*

On the first ten iterations the graph decrease rapidly, which implies that the algorithm was able to find open cells that decrease the objective function value within a few iterations. Few more cells were visited without any improvement to the solution but with luck afterwards. The constant trend, from iteration 45 to iteration 110, implies that the algorithm performed some iterations without any luck of improvement until all the cells were visited. An increasing peak is observed (as expected) when a worsening move was made after all cells have been evaluated without any improvement. When cells are being evaluated, the solution costs are recorded and the cell that worsens the solution the least will be accepted. Accepting the worst solution assisted in escaping the local optimum and an optimal solution was obtained. Thus, after accepting the worsening solution an optimal solution was obtained.

5.4 Lingo performance analysis

Thirty test problems (Dataset 1) along with their outputs are considered in this performance analysis. All test problems have similar dimensions (size 15×15) with different parameters, as discussed in Section 3.1. Table 5.6 shows the results obtained for Dataset 1, where the first column shows the test problem number, the second and the third columns show the the optimal solution and the CPU time (in seconds). Lingo solved the test problems by branch and bound methodology and managed to obtain optimal solution to all the test problems.

In some test problems, optimal solutions was obtained within a short period of time (i.e. Problem 9 of Set A1 and Set A2, Problem 2 of Set A3), whereas optimal solutions was obtained after

	Set A1		Set A2		Set A3	
	Optimal	Time	Optimal	Time	Optimal	Time
1	6683	10.00	10017	5.01	14161	2.15
2	6903	12.08	10075	1.47	13793	0.06
3	6210	0.48	9327	0.18	13699	0.55
4	7753	65.35	11093	117.31	15246	32.19
5	7360	29.57	10312	1.43	14593	8.01
6	6911	0.40	10086	0.16	14680	1.01
7	6434	2.33	9913	2.21	14255	1.25
8	7254	41.09	10495	32.52	14235	4.08
9	7119	0.15	10137	0.04	14281	1.40
10	6843	12.33	10137	1.31	13953	1.44
Average		17.38		16.16		5.21

TABLE 5.6: *Summary of optimal solutions obtained by Lingo solver on Dataset 1*

a long period of time in other instances (e.g. Problem 4 of Set A1, Set A2 and Set A3). The average time taken to obtain optimal solutions to all test problems is 17.38 for Set A1 instances, 16.16 for Set A2 instances and 5.21 for Set A3. On average, optimal solutions to Set A3 was obtained faster that Set A1 and A2. Using LINGO solver, since it uses the branch and bound methodology, the computational time grows along with the problem size. However, LINGO is not capable to provide optimal solutions in large problem instances. The solver time grows rapidly with an increase in the problem dimensions.

5.5 Experimental analysis on Dataset 2

Tables A.6–A.11 in Appendix 1 compare the performance of the Branch and Price (B&P) algorithm developed by Roberti *et al.* [33] and the tabu search algorithm proposed in this thesis based on Dataset 2. These tables report the instance name ("Inst"), the upper bound (UB) and computational time (T_{Tot}) obtained by Roberti *et al.*. For the tabu search algorithm, the table report the best objective function value (Best) and the computational time (T_{Tot}) for solving the instances. The last column report the solution gap to the UB obtained by Roberti *et al.*. Averages are reported in the last lines of each table. The solution obtained by Roberti *et al.* is so far the best known solution and in almost all the problem instances, the B&P outperformed our algorithm on both the solution quality and the computational time. Roberti *et al.* set a time limit of 3 hours on these datasets whereas a time limit was not set when running the proposed algorithm. A limit of 500 iterations in each problem instance was set and the time recorded to get to that solution. The results are summarised in Table 5.7 and Table 5.8.

Table 5.7 shows that our algorithm managed to obtain an average of 16.44% solution gap to the best know solution within an average of 43.16 seconds of computational runtime. This analysis is based Set R1 of Dataset 2. On Set R2 of Dataset 1 the algorithm obtained an average of 16.35% solution gap to the best known solution within and average od 127.00 seconds of computational time. Lastly, on Set R3 of Dataset 2 the algorithm obtained an average of 15.92% solution gap to the best known solution within 231.00 seconds of computational time. Although the solution quality is worse that Roberti *et al.* [33], these solution times are much shorter.

Further computational experiments were performed on Set R4, Set R5 and Set R6. On Set R4, our algorithm obtained an average of 12.78% solution gap to the best know solution within an

average of 425.01 seconds of computational runtime. On Set R5 the algorithm obtained its best solution with an average of 12.78% solution gap to the best know solution within an average of 1296.32 seconds of computational time. This was outperformed by the branch-and-bound algorithm which obtained its solution within 95.2 seconds of computational time (as shown in Table A.7). Lastly, on Set R6 our algorithm obtained an average of 12.65% solution gap to the best know solution. This was obtained within 524.40 seconds of computational runtime as compared to 1534.6 seconds by the branch-and-bound method.

	Set R1 - R3		Set R4 - R6		Set R7 - R9	
Inst.	GAP(%)	T_{Tot}	GAP(%)	T_{Tot}	GAP(%)	T_{Tot}
30×30	16.44	430.16	425.01	11.98	23.39	21.17
50×50	15.36	127.00	12.78	1296.32	10.95	159.62
70×70	15.92	231.00	12.65	524.40	10.19	427.25

TABLE 5.7: *Table showing the average solution gap and average computational runtime on Set R1 to R9 of Dataset 2.*

The last two columns of Table 5.7 presents the computational experiments on Set R7, Set R8 and Set R9 of Dataset 2. The bigger the value of theta $\theta = 0.5$, the higher the costs. Thus, the best known objective function values to this problems is expected to be higher than that of the instances where $\theta = 0.0$ and 0.2. The proposed algorithm obtained an average of 12.39% solution gap to the best known solutions with 21.17 seconds of computational time, whereas the best known solutions was obtained within an average of 9 seconds of computation runtime (as shown in Table A.8). Performing the computational experiments on Set R8 of Dataset 2, the algorithm managed to get an average of 10.95% solution gap to the best known solutions within an average of 159.62 seconds. Although our algorithm was outperformed in terms of solution quality, it performed better than the B&P method which took 324 seconds average of computational time to find its best solutions. Lastly, analysis was ran on Set R9. In this set, our algorithm obtained an average of 10.19% solution gap to the best known solution within 427.25 seconds of computational time. The algorithm also took less computational runtime than the B&P, which took an average of 1395 seconds of computational time.

Table 5.8 shows a summary of the average solution gap to the best known solution. In this computations, the results are based on Set R10 to Set R18 on Dataset 2. On Set R10, the algorithm obtained an average of a 11.79% solution gap to the best know solution within 33.76 seconds average computational runtime; 11.57% solution gap within 25.36 seconds of computational runtime on Set R11, and lastly a 7.98% solution gap within 28.72 seconds of computational runtime on Set R12 of Dataset 2.

	Set R10 - R12		Set R13 - R15		Set R16 - R18	
Inst.	GAP(%)	T_{Tot}	GAP(%)	T_{Tot}	GAP(%)	T_{Tot}
20×20	11.79	33.76	13.95	122.68	21.39	140.73
30×30	11.57	25.36	15.26	96.58	14.93	167.50
50×50	7.98	28.72	9.89	235.18	10.37	232.116

TABLE 5.8: *Table showing the average solution gap and average computational runtime on Set R10 to Set R18 of Dataset 2.*

Columns 4 and 5 shows the solution gap relative to the best known solution and the average

computational runtime taken to obtain the solutions on Set R13, Set R14 and Set R15. The algorithm obtained an average solution gap of 13.95% within 122.68 seconds of computational runtime on Set R13; 15.26% solution gap within 96.58 seconds on Set R14 and 9.89% solution gap within 235.18 seconds of computational runtime on Set R15. On the last set of instances, Set R16, Set R17 and Set R18, the algorithm obtained 21.39%, 14.93% and 10.37% solution gap within 140.73, 167.50 and 232.116 seconds of computational time, respectively. Although the algorithm have not obtained better solution than the B&P algorithm (as presented in Table 5.8), it could achieve good results in less time.

5.6 The primal-dual algorithm solution experiments

The primal-dual method for the transportation problem was applied on the same problem instances as above. This method starts by determining the feasible cells at each iteration and assigning possible allocations to the cells until all supply and demand constraints are satisfied. This method is normally applied on standard transportation problems, which only contains the variable cost on each cell (route). Thus, Balinski relaxation was considered to trade-off the variable cost and the fixed cost to a standard unit cost. When a feasible solution is attained, it is assumed to be an initial feasible solution.

5.6.1 Initial feasible solution

Table 5.9 presents the initial feasible solution to Dataset 1 using the primal-dual method to the fixed charge transpiration problems. The table shows the problem instance (Inst.) on the first column, the initial solution costs (Init.) and the computational time (T_{Tot}) for all sets of instances. The average computational time is presented on the last line of the table. Although the algorithm did not obtain the initial solution to all instances of Dataset 1, it managed to attain a good solution that is near optimal solutions to at least 76% of Dataset 1. Again, the solutions was attained within a short computational time. For 9 instances on Set A1, the algorithm attained the solutions within an average 0.033 seconds of computational runtime, 0.032 seconds for Set A2 and 0.041 seconds of computational runtime for Set A3 of Dataset 1.

5.6.2 Solution improvement

The stepping stone method was applied and the tabu search algorithm to improve the current initial solution obtained. The improvement process was started by sequentially searching for cells that make an improvement to the current solution. This search is applied only on open cells (unallocated cells). Should there exists any cell that makes an improvement, its solution is accepted and the search is restarted. If all the open cells are evaluated without getting a cell that improves the solution, the least worse move is made. The *least worse move* refers to making an allocation to a cell that worsens the solution to the least. This process helps the algorithm to escape the local optimum.

If at each iteration, a cell is evaluated and it makes no improvement to the current solution, its path (if it exists) is stored in a tabu list to avoid being incorporated immediately after a worsening move is made. This list accommodates a maximum of 5 paths, so as to allow old paths to be implemented as soon as they are removed from the tabu list. Table 5.10 presents the best solutions to Dataset 1, obtained using the stepping stone algorithm in conjunction with the tabu search algorithm. This methods were applied to the initial solutions obtained using

	Set A1		Set A2		Set A3	
Inst.	Init.	T_{Tot}	Init.	T_{Tot}	Init.	T_{Tot}
1	8456	0.22	11367	0.06	–	–
2	8985	0.05	12080	0.04	16519	0.03
3	8148	0.03	10133	0.02	16106	0.03
4	8870	0.04	–	–	17668	0.06
5	8255	0.03	11915	0.03	17297	0.05
6	8350	0.02	11436	0.03	17223	0.05
7	7149	0.03	–	–	–	–
8	8916	0.03	–	–	16472	0.03
9	–	–	11305	0.03	15405	0.04
10	8238	0.03	11165	0.03	15736	0.03
Average		0.033		0.032		0.041

TABLE 5.9: *Initial solution to Dataset 1 instances of the fixed charge transportation problems using the primal-dual method.*

the primal-dual methods. Although optimal solutions has not been attained to this problems instances, the algorithm managed to get near optimal solution with an average of 6.77%, 3.86% and 4.45% solution gap to Set A1, Set A2 and Set A3 problem instances, respectively. This method appears to be faster than all methods tested in this thesis as it attained its best solution within an average of 0.47%, 0.22% and 0.22% computational runtime with Set A1, Set A2 and Set A3 problem instances, respectively.

	Set A1				Set A2				Set A3			
Inst.	Opt.	Best	GAP	T_{TOT}	Opt.	Best	GAP	T_{TOT}	Opt.	Best	GAP	T_{TOT}
1	6 683	6972	3.32	0.22	10 017	10348	3.30	0.16	14 161	–	–	
2	6 903	7709	11.68	1.07	10 075	10455	3.77	0.18	13 793	14820	7.45	0.11
3	6 210	6589	6.10	0.23	9 327	9523	2.10	0.13	13 699	14178	3.47	0.41
4	7 753	7828	0.97	0.62	11 093	–	–	–	15 246	16160	5.99	0.46
5	7 360	7602	3.29	0.33	10 312	11044	7.10	0.40	14 593	14739	1.00	0.21
6	6 911	7754	12.20	0.62	10 086	10402	3.13	0.07	14 680	15489	5.51	0.05
7	6 434	6577	2.22	0.12	9 913	–	–	–	14 255	–	–	–
8	7 254	7830	7.94	0.64	10 495	–	–	–	14 235	14971	5.17	0.26
9	7 119	–	–	–	10 137	10429	2.88	0.32	14 281	14774	3.45	0.16
10	6 843	7273	6.28	0.31	9 939	10411	5.33	0.24	13 953	14441	3.50	0.13
Average			6.11	0.47			3.86	0.22			4.45	0.22

TABLE 5.10: *Best solution to Dataset 1 using the stepping stone method on primal-dual method solutions to the fixed charge transportation problems.*

The tabu search algorithm was applied in different ways in this algorithm. Table 5.11 also depicts the best solution costs obtained using the stepping stone method and tabu search algorithm. Every open cell is evaluated for solution improvement and if there exists a cell that makes an improvement to the current solution cost, its solution is accepted. A least worsening move is made if all open cells have been evaluated without any improvement. Should a solution be accepted (better or worse), at least one cell will leave the basis. The cell that leaves the basis is

stored in a tabu list to avoid an immediate cycle. The size of the tabu list is set to 10 by default, and if more cells leave the basis; the first cell that entered the list is removed. This method obtained near optimal solution with an average of 10.02%, 4.89% and 4.98% solution gap to Set A1, Set A2 and Set A3 problem instances, respectively. The algorithm obtained its best solution within an average of 0.16, 0.12 and 0.14 computational time for the above respective instances. Computationally, the algorithm is faster than most tested algorithms in the thesis although it did not obtain optimal solution.

	Set A1				Set A2				Set A3			
Inst.	Opt.	Best	GAP	T_{TOT}	Opt.	Best	GAP	T_{TOT}	Opt.	Best	GAP	T_{TOT}
1	6 683	6972	3.32	0.22	10 017	10348	3.30	0.16	14 161	–	–	–
2	6 903	8088	17.17	0.23	10 075	10708	6.28	0.13	13 793	14820	7.45	0.09
3	6 210	7575	21.98	0.14	9 327	9523	2.10	0.13	13 699	14390	5.04	0.14
4	7 753	8124	4.79	0.16	11 093	–	–	–	15 246	16477	8.07	0.20
5	7 360	8064	9.57	0.13	10 312	11380	10.36	0.13	14 593	14831	1.63	0.13
6	6 911	8075	16.84	0.08	10 086	10402	3.13	0.08	14 680	15489	5.51	0.03
7	6 434	6577	2.22	0.14	9 913	–	–	–	14 255	–	–	–
8	7 254	7991	10.16	0.16	10 495	–	–	–	14 235	14971	5.17	0.23
9	7 119	–	–	–	10 137	10501	3.59	0.14	14 281	14774	3.45	0.16
10	6 843	7434	8.64	0.16	9 939	10469	5.33	0.11	13 953	14441	3.50	0.14
Average			10.02	0.16			4.87	0.12			4.98	0.14

TABLE 5.11: *Best solution to Dataset 1 problem instances using the stepping stone method on primal-dual method solutions to the fixed charge transportation problems.*

5.7 Chapter summary

The best obtained initial feasible solution was improved using the stepping stone method, considering the greedy local search algorithm and the tabu search algorithm. Optimal and near optimal solutions was obtained and presented in this chapter. When evaluating cells for solution improvement, several selection criteria were proposed and the criterion with better results was considered.

Furthermore, the result from the primal-dual for solving the fixed charge transportation problems was presented. The algorithm was only based on Dataset 1. The algorithm managed to obtain feasible solutions to most of the problem instances. The obtained feasible solutions was considered to be initial solutions, and were improved using the tabu search algorithm in conjunction with the stepping stone method. Although the algorithm failed to obtain optimal solutions, near optimal solutions were obtained with a reasonable computational runtime.

CHAPTER 6

Conclusion and recommendations

Contents

6.1 Thesis summary 53
6.2 Recommendation 54
6.3 Achievement of objectives 54
6.4 Future work 54

6.1 Thesis summary

In this thesis, an extension of the classical transportation problem called the *fixed charge transportation problem* was investigated. In this extension a fixed cost on an arc is incurred, independent of the amount transported on that arc, along with the normal variable cost that is proportional to the amount shipped. This study is based on the objective of determining which routes are to be opened and the amount of the shipment on those routes, so that the total cost of meeting demand, given the supply constraints, is minimised. This research could be extended in a number of ways to attempt to improve the computational efficiency of the fixed charge transportation algorithms.

An efficient heuristic procedure for solving a fixed charge transportation problem was introduced. The results of this study indicate some characteristics of Balinski's technique of relaxation the problem by a trade off to the variable costs and the fixed costs so as to determine the initial solutions. Benchmark instances from the literature was used to test the performance of this heuristic methods. Several experiments were made based on different criteria, to determine a good initial solution. The stepping stone method to improve the initial solution was described and also a tabu search algorithm is considered.

The primal-dual method for solving transportation problem was also discussed and developed in this thesis. This method approach the problem by searching feasible cells to the problem and allocating possible commodity without considering any primal feasibility. An important benefit of this method is that degeneracy causes no challenge to this method. Once a feasible solution is obtained, the stepping stone with tabu search algorithm is considered in an attempt to improve the solution.

6.2 Recommendation

A the Tabu search algorithm and the stepping stone method was employed to solve the nonlinear fixed charge transportation problem, which was developed based on the solution structure for the linear transportation problem. The proposed algorithm could not obtain optimal solutions to all the tested problems. It is recommended that the tabu search algorithm based on the stepping stone method must be further explored and improved when solving the nonlinear fixed charge transportation problem. As this methods shows the most promise.

6.3 Achievement of objectives

Exact methods and different classes of heuristics has been studied from the literature on the fixed charge transportation problems. Benchmark instances from the literature was used to evaluated the heuristics based on both computational runtime (in seconds) and solution quality. To investigate the computational efficiency of the proposed methods, the instances used by Roberti *et al.* [33] based on Dataset 1 and Agarwal and Aneja [5] based on Dataset 2 was used. The proposed heuristic approaches delivered near optimal solution to most of the tested instances with one only problem instance solving to optimality. The primal-dual algorithm demonstrate significant improvement over the proposed heuristic methods for small fixed charge transportation problems.

6.4 Future work

In terms of methodology future research might focus on further refinement of both the tabu search as well as the primal-dual algorithm. Both these methods might be improved by making changes to their functioning. Other neighbourhoods might for example be considered in the tabu search algorithm.

For future research other more general assumptions may also be considered, such as a dynamic environment and truck availability constraints. Aspects that can be done in future may also be a need to include more variables for each transport mode, such as: time of the trip; consumption of energy; emission of pollutants to the environment; associated costs to these factors. These variables can be assumed as macroscopic by doing estimations or analysing statistical data. Macroscopic variables can be defined as costs associated with staff engaged in the activity; delivery deadlines in order to create different priorities of visit to each customer. This might lead to a multi-objective approach to FCTP.

APPENDIX A

Computational results

This chapter presents the detailed results to the computational experiments of the datasets made in this thesis. Table A.1 shows the initial solutions for the FCTP when the Balinski's relaxation is considered. Four different selection criteria (i.e. RAM, ReMCM, RMCM and MaxFlowMCM) were considered and evaluated based on the computation runtime and solution quality. Dataset 1 of the benchmark instances were used to test the algorithms. Dataset 2 and Dataset 3 were also considered to test the performance of the same selection criteria and shown in Table A.2 and Table A.3, respectively.

The better performing selection criterion was found to be the MaxFlowMCM and further experiments were made on this algorithm. Dataset 1 was used to improve the current initial solutions using the greedy search algorithm and the tabu search algorithm. The improved solution experiments are presented in Table A.4 and Table A.5. Finally the computational experiments of the of Level 2 datasets and the results are presented in Table A.6–Table A.11.

Problem no.	RAM	T_{Tot}	ReMCM	T_{Tot}	RMCM	T_{Tot}	MaxFlowMCM	T_{Tot}
1	16446	0.028	8665	0.027	9377	0.029	8360	0.026
2	18608	0.029	11042	0.027	10937	0.029	9434	0.025
3	16140	0.029	9262	0.032	9577	0.030	9241	0.029
4	16665	0.029	9801	0.028	12096	0.028	10937	0.028
5	15376	0.030	10271	0.028	11976	0.030	9665	0.027
6	17886	0.029	8828	0.029	11184	0.030	10150	0.027
7	16791	0.029	6958	0.029	10398	0.027	7748	0.025
8	17551	0.028	9591	0.029	11182	0.031	10049	0.027
9	17193	0.028	8838	0.029	13762	0.031	8781	0.026
10	17359	0.027	9506	0.028	9653	0.031	8966	0.027
Average		0.0287		0.0286		0.0269		0.0268

TABLE A.1: *Initial solutions for fixed charge transportation problems when Balinski's relaxation is considered on Set A1 of Dataset 1*

Problem no.	RAM	T_{Tot}	ReMCM	T_{Tot}	RMCM	T_{Tot}	MaxFlowMCM	T_{Tot}
1	21848	0.029	11669	0.029	14093	0.046	10979	0.028
2	20908	0.030	12900	0.028	14216	0.028	13204	0.027
3	20746	0.029	11808	0.032	14282	0.029	10717	0.027
4	20954	0.029	14229	0.030	15071	0.030	13424	0.028
5	19029	0.029	12694	0.029	12953	0.029	12648	0.027
6	16951	0.028	12807	0.030	13331	0.030	11092	0.028
7	18292	0.028	11597	0.028	12847	0.029	11981	0.028
8	19759	0.029	13356	0.029	13482	0.030	13692	0.027
9	21822	0.030	13042	0.030	15353	0.031	13982	0.023
10	20321	0.028	12059	0.031	13746	0.030	12703	0.030
Average		0.0290		0.0296		0.0312		0.0273

TABLE A.2: *Initial solutions for fixed charge transportation problems when Balinski's relaxation is considered Set A2 of Dataset 1*

Problem no.	RAM	T_{Tot}	ReMCM	T_{Tot}	RMCM	T_{Tot}	MaxFlowMCM	T_{Tot}
1	25979	0.029	15169	0.029	18900	0.027	14779	0.027
2	25238	0.029	16785	0.029	17292	0.030	16017	0.027
3	25817	0.029	14985	0.030	17178	0.030	16218	0.028
4	24089	0.030	18738	0.028	19645	0.027	17824	0.026
5	24883	0.029	18696	0.029	19641	0.030	17788	0.027
6	28629	0.028	16288	0.031	19266	0.030	16646	0.028
7	23991	0.028	18240	0.030	17330	0.026	16545	0.028
8	25785	0.047	17055	0.028	18073	0.031	16780	0.027
9	28020	0.046	15673	0.030	18144	0.030	16172	0.027
10	25708	0.029	16240	0.030	18363	0.030	16695	0.030
Average		0.0324		0.0294		0.0291		0.0275

TABLE A.3: *Initial solutions for fixed charge transportation problems when Balinski's relaxation is considered on Set A3 of Dataset 1*

	Set A1			Set A2			Set A3		
Problem No.	Opt.	Best	T_{Tot}	Opt.	Best	T_{Tot}	Opt.	Best	T_{Tot}
1	6680	7905	0.099	10017	10734	0.058	14161	14974	0.068
2	6903	8056	0.106	10075	12104	0.196	13793	15467	0.060
3	6210	7837	0.044	9327	11387	0.059	13699	15206	0.073
4	7753	9780	0.113	11093	12736	0.132	15246	16164	0.247
5	7360	9045	0.061	10312	12432	0.054	14593	15801	0.081
6	6911	7672	0.196	10086	11506	0.038	14680	16674	0.075
7	6434	7837	0.069	9913	11519	0.106	14255	15337	0.190
8	7254	8978	0.097	10495	11990	0.281	14235	15371	0.185
9	7119	8478	0.048	10137	11281	0.067	14281	16442	0.137
10	6843	7634	0.136	9939	10971	0.198	13953	14468	0.090
Average			0.097			0.119			0.121

TABLE A.4: *Best FCTP solutions using the MaxFlowMCM to the greedy search algorithm on Dataset 1*

	Set A1			Set A2			Set A3		
Problem No.	Opt.	Best	T_{Tot}	Opt.	Best	T_{Tot}	Opt.	Best	T_{Tot}
1	6680	7136	2.67	10017	10231	0.20	14161	14351	3.13
2	6903	7798	2.11	10075	11157	0.74	13793	15144	0.29
3	6210	6608	5.21	9327	9981	4.20	13699	14234	1.69
4	7753	8752	0.80	11093	11408	0.73	15246	16164	0.21
5	7360	8332	0.89	10312	11567	1.09	14593	15432	0.27
6	6911	7342	3.61	10086	10393	3.97	14680	15361	4.51
7	6434	7727	0.42	9913	11566	2.98	14255	14255	2.71
8	7254	8138	2.51	10495	11258	2.46	14235	14926	2.13
9	7119	8153	4.57	10137	10759	0.24	14281	14935	4.80
10	6843	7488	0.19	9939	7488	0.19	13953	14468	0.11
Average			2.31			1.681			1.981

TABLE A.5: *Best FCTP solutions using the MaxFlowMCM on Dataset 1, when the tabu search algorithm is considered*

	B&P		Tabu		
Inst	UB	T_{Tot}	Best	T_{Tot}	GAP (%)
30×30					
1	10690	4	11611	167.52	7.93
2	10443	5	11708	54.50	10.80
3	10918	6	12359	0.21	11.66
4	11365	7	13173	182.78	13.73
5	10543	21	12872	105.56	18.09
6	10799	10	11921	101.00	9.41
7	10939	22	13222	216.73	17.27
8	10588	7	14328	0.22	26.10
9	10558	39	11509	90.93	8.26
10	10747	7	11968	256.76	10.20
Avg		13		117.62	13.35
50×50					
1	15972	121	18329	514.45	12.86
2	16154	42	18458	581.36	12.48
3	15996	35	18474	1374.86	13.41
4	16317	116	19295	1371.71	15.43
5	16147	151	18472	977.28	12.59
6	16576	341	18803	995.27	11.84
7	15854	87	18269	763.83	13.22
8	16043	1533	19570	772.52	18.02
9	16326	76	18778	1.83	13.06
10	15898	34	18968	383.35	16.19
Avg		116		773.65	13.91
70×70					
1	21155	896	24297	1053.27	12.93
2	21614	211	26929	3.34	19.74
3	21346	1154	25106	540.61	14.98
4	20771	528	24859	1576.59	16.44
5	21107	179	25329	1343.61	16.67
6	20343	194	23800	1414.50	14.53
7	21033	288	24829	857.04	15.29
8	20816	275	23437	4.95	11.18
9	21123	320	25369	52.83	16.74
10	21010	162	24495	1272.26	14.23
Avg		421		811.90	15.27

TABLE A.6: *Computational results on Set R1 to Set R3 of Dataset 2.*

	B&P		Tabu		
Inst	UB	T_{Tot}	Best	T_{Tot}	GAP (%)
30 × 30					
1	12769	3	14142	514.26	9.71
2	12979	3	14637	504.21	11.33
3	14109	7	15759	508.39	10.47
4	13217	9	14772	589.77	10.53
5	13756	11	17638	1.30	22.01
6	13540	18	15022	520.55	9.87
7	13547	11	14673	514.79	7.67
8	13116	36	15496	1.28	15.36
9	13836	9	15090	497.11	8.31
10	13371	6	15654	598.48	14.58
Avg		11.3		425.01	11.98
50 × 50					
1	20451	69	23129	1429.50	11.58
2	20704	82	23037	1450.99	10.13
3	20672	154	23738	1525.16	12.92
4	20757	97	23812	1371.05	12.83
5	21097	159	24091	1463.21	12.43
6	20751	32	23097	1435.57	10.16
7	20475	50	23401	1438.41	12.50
8	20927	169	24998	1450.00	16.29
9	20903	92	23221	1395.27	9.98
10	20320	48	25082	4.09	18.99
Avg		95.2		1296.32	12.78
70 × 70					
1	27868	5854	32363	569.87	13.89
2	27087	443	30879	587.89	12.28
3	27547	390	30511	580.70	9.71
4	26832	90	30782	8.56	12.83
5	27685	228	31006	592.74	10.71
6	26972	206	31447	581.11	14.23
7	27485	1171	31620	600.16	13.08
8	26944	6528	32504	566.47	17.11
9	27769	267	31119	582.30	10.77
10	27256	169	30949	574.19	11.93
Avg		1534.6		524.40	12.65

TABLE A.7: *Computational results on Set R4 to Set R6 of Dataset 2.*

	B&P		Tabu		
Inst	UB	T_{Tot}	Best	T_{Tot}	GAP (%)
30×30					
1	18291	7	20809	0.23	12.10
2	19106	5	20532	79.62	6.95
3	18034	5	23031	0.22	21.70
4	17637	4	20297	0.25	13.11
5	18548	11	20712	10.57	10.45
6	17781	19	20754	0.24	14.32
7	17969	9	19841	10.29	9.44
8	18198	10	20541	70.23	11.41
9	17744	7	19644	39.78	9.67
10	18760	8	22016	0.23	14.79
Avg		9		21.17	12.39
50×50					
1	27147	107	30220	17.05	10.17
2	27574	209	32930	1.07	16.26
3	27668	200	30379	296.46	8.92
4	27603	1207	31418	1.06	12.14
5	27085	1170	29441	117.00	8.00
6	28256	121	30651	333.49	7.81
7	27493	72	31711	1.07	13.30
8	27621	30	29376	401.54	5.97
9	27445	45	30934	426.43	11.28
10	27888	76	33062	1.02	15.65
Avg		324		159.62	10.95
70×70					
1	36584	477	42383	2.88	13.68
2	37551	2349	42809	2.77	12.28
3	36982	585	40111	3.02	7.80
4	36669	4657	41199	830.55	11.00
5	37090	1946	41341	760.53	10.28
6	37152	342	40259	2.93	7.72
7	36715	1660	40872	592.06	10.17
8	37007	201	40828	784.02	9.36
9	37679	1546	41400	798.77	8.99
10	36588	182	40914	494.97	10.57
Avg		1395		427.25	10.19

TABLE A.8: *Computational results on Set R7 to Set R9 of Dataset 2.*

	B&P		Tabu		
Inst	UB	T_{Tot}	Best	T_{Tot}	GAP (%)
20×20					
1	7710	3	8199	40.871	5.96
2	8039	5	10441	0.066	23.01
3	8483	121	9205	21.280	7.84
4	8223	94	9246	17.118	11.06
5	8257	9	9492	82.363	13.01
6	8684	63	10924	0.065	20.51
7	8576	3	9515	44.540	9.87
8	8329	64	9404	42.477	11.43
9	7863	5	8238	32.304	4.55
10	8548	6	9566	56.463	10.64
Avg		37		33.755	11.79
30×30					
1	11513	36	12713	98.024	9.44
2	11649	329	13995	155.878	16.76
3	11144	242	12726	125.910	12.43
4	11303	280	13748	108.675	17.78
5	11543	290	13288	10.895	13.13
6	11642	1449	13716	78.216	15.12
7	11400	54	13873	173.495	17.83
8	10817	145	12137	94.241	10.88
9	11911	121	13829	166.299	13.87
10	11259	585	12835	215.142	12.28
Avg		353		122.678	13.95
40×40					
1	14285	3798	17336	368.321	17.60
2	14510	899	19281	0.514	24.74
3	14409	6638	17612	335.310	18.19
4	14651	6840	20842	0.522	29.70
5	14732	9456	16945	244.045	13.06
6	14348	7289	20159	0.515	28.83
7	14785	9341	17379	204.300	14.93
8	13615	4510	17792	0.521	23.48
9	14408	8103	18597	0.522	22.53
10	14169	8411	17896	252.731	20.83
Avg		5060		140.73	21.39

TABLE A.9: *Computational results on Set R10 to Set R12 of Dataset 2.*

	B&P		Tabu		
Inst	UB	T_{Tot}	Best	T_{Tot}	GAP (%)
20 × 20					
1	10286	45	13817	0.063	25.56
2	10610	5	11649	44.766	8.92
3	10746	19	11347	30.178	5.30
4	10769	3	11870	24.311	9.28
5	10521	19	11354	34.716	7.34
6	9802	11	11078	29.618	11.52
7	9337	14	11975	0.064	22.03
8	10562	25	11294	37.132	6.48
9	10411	15	12246	0.065	14.98
10	9947	4	10392	52.706	4.28
Avg		16		25.36	11.57
30 × 30					
1	13969	2886	17480	0.224	20.09
2	14310	188	16088	164.613	11.05
3	13707	1250	17236	0.212	20.47
4	14482	2487	16061	77.935	9.83
5	13888	519	15363	137.614	9.60
6	13822	116	15745	197.815	12.21
7	14551	808	16622	211.681	12.46
8	14039	470	15218	175.219	7.75
9	14079	94	18266	0.219	22.92
10	14537	173	19693	0.228	26.18
Avg		899.1		96.58	15.26
40 × 40					
1	17931	9256	20548	236.286	12.74
2	18249	9510	21308	242.043	14.36
3	17738	8888	20019	366.433	11.39
4	17911	364	21001	0.532	14.71
5	17300	1568	21788	0.533	20.60
6	17815	3065	21249	378.945	16.16
7	17727	9178	23302	0.537	23.92
8	18024	1853	20625	120.405	12.61
9	18428	1489	20803	47.456	11.42
10	17900	2176	20202	281.836	11.39
Avg		4734.7		167.50	14.93

TABLE A.10: *Computational results on Set R13 to Set R15 of Dataset 2.*

	B&P		Tabu		
Inst	UB	T_{Tot}	Best	T_{Tot}	GAP (%)
20 × 20					
1	13335	5	13684	57.798	2.55
2	12947	4	14557	0.067	11.06
3	13936	23	14371	79.183	3.03
4	13191	5	13650	10.652	3.36
5	13759	15	14116	58.404	2.53
6	13774	1	18376	0.076	25.04
7	13184	5	13800	33.339	4.46
8	13134	14	13751	32.406	4.49
9	12811	50	15410	0.083	16.87
10	13218	9	14125	15.208	6.42
Avg		13.1		28.722	7.98
30 × 30					
1	18298	148	21748	0.220	15.86
2	18785	153	20960	414.989	10.38
3	18844	115	23818	0.233	20.88
4	18163	768	19000	160.791	4.41
5	18628	665	19264	490.903	3.30
6	18922	234	20201	335.089	6.33
7	18353	474	19352	383.166	5.16
8	18950	420	23541	0.214	19.50
9	18036	213	19040	140.516	5.27
10	19068	675	20687	425.694	7.83
Avg		386.5		235.182	9.89
40 × 40					
1	23480	10704	28226	0.523	16.81
2	23282	629	25886	152.147	10.06
3	23927	135	27205	415.319	12.05
4	24858	10027	26605	345.869	6.57
5	23486	9007	25830	268.099	9.07
6	23781	9678	26504	211.908	10.27
7	23914	4460	26459	169.655	9.62
8	23804	8214	25968	203.347	8.33
9	23989	10733	27622	214.181	13.15
10	23950	8296	25957	340.110	7.73
Avg		7188.3		232.116	10.37

TABLE A.11: *Computational results on Set R16 to Set R18 of Dataset 2.*

Bibliography

[1] Adlakha V and Kowalski K, 1999, *On the fixed-charge transportation problem*, The International Journal of Management Science, **27(3)**, pp. 381–388.

[2] Adlakha V and Kowalski K, 2013, *A simple heuristic for solving small fixed-charge trasoportation problems*, Omega, **31(3)**, pp. 205–211.

[3] Adlakha V, Kowalski K and Lev B, 2010, *A branching method for the fixed charge transportation problem*, Omega, **38(5)**, pp. 393–397.

[4] Adlakha V, Kowalski K, Wang S, Lev B and Shen W, 2014, *On approximation of the fixed charge transportation problem*, **43(C)**, pp. 64–70.

[5] Agarwal Y and Aneja Y, 2012, *Fixed charge transportation problem: Facets of the projection polyhendron*, Operations Research, **60(3)**, pp. 638–654.

[6] Aguado J, 2009, *Fixed charge transportation problems: a new heuristic approach based on lagrangean relaxation and solving of core problems*, Annals of Operations Reseasrch, **172(1)**, pp. 45–69.

[7] Altassan K, El-Sherbiny M and Abid A, 2014, *Articial immune algorithm for solving fixed charge transportation problem*, Applied Mathematics & Information Science, **8(2)**, pp. 751–759.

[8] Balinski M, 1961, *Fixed cost transportation problems*, Naval Research Logistics Quaterly, **8(1)**, pp. 41–54.

[9] Brandao J, 2004, *A tabu search algorithm for the open vehicle routing problem*, European Journal of Operational Research, **157(3)**, pp. 552 – 564.

[10] Brownlee J, *Clever algorithms: Nature-inspired programming recipes*, Available from `\url{http://www.cleveralgorithms.com/nature-inspired/stochastic/tabu_search.html}`, accessed: 2016-09-30.

[11] Buson E, Roberti R and Toth P, 2014, *A reduced-cost iterated local search heuristic for the fixed-charge transportatopn problem*, Operations Research, **62(5)**, pp. 1095–1106.

[12] Cooper L and Steinberg D, 1974, *Methods and Applications of Linear Programming.*

[13] Dantzig G, Ford L and Fulkerson D, 1956, *A primal-dual algorithm for linear programs: In linear inequalities and related systems*, Annals of Mathematics Studies, **38(2)**, pp. 171–181.

[14] Dreo J, Petrowski A and Taillard E, 2003, *Metaheuristics for hard optimization: Simulated annealing, tabu search, evolutionary and genetic algorithms, ant colonies*, Methods and Case Studies.

[15] Gen M and Altiparmak L, 2006, *A genetic algorithm for two-stage transportation problem using priority-based encoding*, Operations Research Spectrum, **28(3)**, pp. 337–354.

[16] Glover F, Amini M and Kochenberger G, 2005, *Parametric ghost image processes for fixed-charge problems: a study of transportation networks*, Journal of Heuristics, **11(4)**, pp. 307–336.

[17] Gray P, 1971, *Exact solution of the fixed-charge transportation problem*, Management Science, **19(6)**, pp. 1529–1538.

[18] Hajiaghaei-Keshteli M, Molla-Alizadeh-Zavardehi S and Tavakkoli-Moghaddam R, 2010, *Addressing a nonlinear fixed-charge transportation problem using a spanning treee-based genetic algorithm*, Computers & Industrial Engeneering, **59(2)**, pp. 259–271.

[19] Hirsch W and Dantzig G, 1968, *The fixed charge problem*, Naval Research Logistics, **15(3)**, pp. 413–424.

[20] Hooker J, 2012, *Towards unification of exact and heuristic optimization methods*, Carnegie Mellon University, **22(1)**, pp. 1–35.

[21] Hultberg T and Cardoso D, 1997, *The teacher assignment problem: A special case of the fixed charge transportation problem*, European Journal of Operations Research, **101(3)**, pp. 463–473.

[22] Kowalski K, 2005, *On the structure of the fixed charge transportation problem*, International Journal of Mathematical Education in Science and Technology, **36(8)**, pp. 879–888.

[23] Kowalski K and Lev B, 2008, *On step fixed-charge transportation problem*, The international Journal of Management Science, **36(5)**, pp. 913–917.

[24] Kowalski K, Lev B, Shen W and Tu Y, 2014, *A fast and simple branching algorithm for solving small scale fixed-charge transportation problem*, Operations Research Perspectives, **1(1)**, pp. 1–5.

[25] Lotfi M and Moghaddam R, 2013, *A genetic algorithm using prority-base encoding with new operators for fixed charge transportation problems*, Applied Soft Computing, **13(5)**, pp. 2711–2726.

[26] Luke S, 2009, *Essentials of metaheuristics*, Lulu, available at: `http://cs.gmu.edu/$\sim$sean/book/metaheuristics/`.

[27] Murty K, 1968, *Solving the fixed charge problem by ranking the extreme points*, Operations Research, **16(2)**, pp. 268–279.

[28] Natarajan A, Balasubramani P and Tamilarasi A, 2006, *Operations Research*, Pearson Education, Singapore.

[29] News A, 2012, *Introduction to algorithms, heuristic and meta-heuristic*, Available from `\url{https://eswarann.wordpress.com/2012/02/12/introduction-to-algorithms-heuristics-and-meta-heuristics/}`, last accessed: 2016-11-09.

[30] Othman Z, Delavar M, Behnam S and Lessanibahri S, 2011, *Adaptive genetic algorithm for fixed-charge transportation problem*, Proceedings of the International MultiConference of Engineers and Computer Scientists.

[31] Ray K and Rajendran C, 2012, *A genetic algorithm for solving the fixed charge transportation model: Two-stage problem*, Computers and Operations Research, **39(9)**, pp. 2016–2032.

[32] Robers P and Cooper L, 1976, *A study of fixed charge transportation problem*, Computers & Mathematics with Applications, **2(2)**, pp. 125–135.

[33] Roberti R, Bartolono E and Mingozzi A, 2014, *The fixed-charge transportation problem: An exact algorithm on a new integer programming formulation*, Management Science, **61(6)**, pp. 1–17.

[34] Roe A and Winston W, *Lindo systems inc. version 17.0*, `https://www.lindo.com/lindoforms/downloadWayne.php`.

[35] Sanei M, Mahmoodirad A, Hassasi H and Rahimian M, 2014, *Fixed-charge transportation problem with fuzzy costs*, Journal of Applied Science and Agriculture, **9(9)**, pp. 1–8.

[36] Stroup J, 1967, *Allocation of launch vehicles to space missions: a fixed-cost transportation problem*, Operations Research, **15(6)**, pp. 1157–1163.

[37] Sun M, Aronson J, McKeown P and Drinka D, 1998, *A tabu search heuristic procedure for the fixed charge transportation problem*, European Journal of Operational Research, **106(2)**, pp. 441–456.

[38] Szabo J, 2016, *Comparison of methods for generating initial solutions for simulated annealing*, Central European Researchers Journal, **2(1)**, pp. 1–41.

[39] Talbi E, Petrowski A and E T, 2009, *Metaheuristics: From design to experiment*, University of Lille – CNRS – INRIA, John Wiley & Sons.

[40] Van Rossum G, *Python programming language, version 3.3.3*, Available at `http://www.python.org`.

[41] Vivtoriano P, 2005, *Quantitative techniques for business management*, Second edition, 1990 revised edition.

[42] Walker W, 1976, *A heuristic adjacent extreme point algorithm for the xed charge problem*, Management Science, **22(5)**, pp. 587–596.

[43] Winston W, 2004, *Operations research: Applications and algorithms*, Thomson-Reuters, Indian University.

[44] Xie F and Jia R, 2012, *Nonlinear fixed charge transportatopn problem by minimum cost flow-based genetic algorithm*, Computers and Industrial Engeneering, **63(4)**, pp. 763–778.

[45] Zionts S, 1974, *Linear and integer programming*, Prentice-hall.